LEARNING TO GENERATE 3D TRAINING DATA THROUGH HYBRID GRADIENT

Anonymous authors
Paper under double-blind review

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 and depth 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 (Butler et al., 2012; Yeh et al., 2012; Varol et al., 2017; Ros et al., 2016; McCormac et al., 2017; Xia et al., 2018; Chang et al., 2017; Kolve et al., 2017; Song et al., 2017; Richter et al., 2016; 2017; Zhang et al., 2017; Li & Snavely, 2018). 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. In fact, recent work by Yang & Deng (2018) has explored this exact direction. They 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.

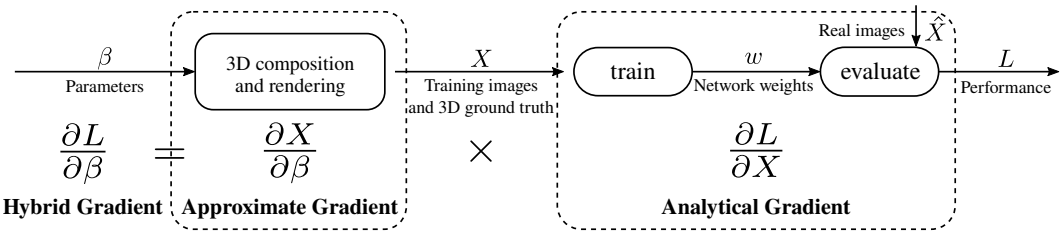

Figure 1: Our "hybrid gradient" method. We parametrize the design decisions as a real vector β and optimize the function of performance L with respect to β. From β to the generated training images and ground truth, we compute the approximate gradient by averaging finite difference approximations. From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

The advantage of black-box optimization is that it makes no assumption 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 good update to the current solution. In contrast, gradient-based optimization can be much more efficient by assuming the availability of analytical gradients, which can be efficiently computed and directly correspond to good updates to the current solution, but the downside is that analytical gradients are 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 analytical gradients where they are available, and combine them with black-box optimization for the rest of the function. Our hypothesis is 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 network performance L can be decomposed into two parts: (1) from 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 gradients, due to the use of advanced photorealistic renderers. We instead compute an approximate gradient by averaging finite difference approximations along random directions (Mania et al., 2018). For the second part, we compute analytical gradients 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 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 & Deng (2018) 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) (Harrison, 1978). 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 our knowledge, we are the first to propose the idea of hybrid gradient, i.e. combining approximate gradients and analytical gradients, especially in the context of optimizing the generation of 3D training data. Second, the integration of PCFG-based shape generation and hybrid gradient is also novel.

We evaluate our approach on the task of estimating surface normal and depth 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.

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 (Song et al., 2015; Hua et al., 2016; McCormac et al., 2017; Janoch et al., 2011; Yang & Deng, 2018; Chang et al., 2015), optical flow estimation (Mayer et al., 2018; Butler et al., 2012; Gaidon et al., 2016), human pose estimation (Varol et al., 2017; Chen et al., 2016), action recognition (Roberto de Souza et al., 2017), natural language modeling (Johnson et al., 2017), and many others (Qiu et al., 2017; Martinez-Gonzalez et al., 2018; Xia et al., 2018; Tobin et al., 2017; Richter et al., 2017; 2016; Wu et al., 2018). The success of these works has demonstrated the effectiveness of synthetic images.

To ensure the relevance of the generated the 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 (Chang et al., 2015; Janoch et al., 2011; Choi et al., 2016; Xiang et al., 2016; Hua et al., 2016; McCormac et al., 2017; Song et al., 2017). To reduce manual labor, some heuristics have been proposed to automatically generate 3D configurations. For example, Zhang et al. (2017) design an approach to use entropy of object masks and color distribution of the rendered image to select sampled camera poses. McCormac et al. (2017) simulate gravity for physically plausible object configurations inside a room.

Prior work has also performed explicit optimization of 3D configurations. For example, Yeh et al. (2012) synthesizes layouts with the target of satisfying constraints such as non-overlapping and occupation. Jiang et al. (2018) learns a probabilistic grammar model for indoor scene generation, with parameters learned using maximum likelihood estimation on the existing 3D configurations in SUNCG (Song et al., 2017). Similarly, Veeravasarapu et al. (2017) tunes the parameters for stochastic scene generation using generative adversarial networks, with the goal of making synthetic images indistinguishable from real images. Qi et al. (2018) 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.

The closest prior work to ours is that of Yang & Deng (2018), who use a genetic algorithm to optimize the 3D shapes used for rendering synthetic training images. Their optimization objective is the same as ours, but their optimization method is different in that they do not use any gradient information. Similarly, Meta-Sim (Kar et al., 2019) also tries to optimize 3D parameters with REINFORCE towards better task generalization performance. However, it does not backpropagate analytical gradients from the meta-objective, so their algorithm is still black-box estimation by mulitple trials.

Unrolling and backpropogating 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 (Maclaurin et al., 2015) and meta-learning (Andrychowicz et al., 2016; Ha et al., 2017; Munkhdalai & Yu, 2017; Li & Malik, 2017; Finn et al., 2018). 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 it is integrated with approximate gradients to form hybrid gradients.

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 is typically approached as black-box optimization (Bergstra & Bengio, 2012; Bergstra et al., 2011; Lacoste et al., 2914; Brochu et al., 2010). Since black-box optimization does not assume knowledge about 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 analytical gradients from backpropagation and approximate gradient from generalized finite difference for more efficient optimization.

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:

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

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:

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

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 repeated evaluation 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 (Jiang et al., 2018; Qi et al., 2018) and in parsing of Constructive Solid Geometry (CSG) shapes (Sharma et al., 2018). Here, we design a probabilistic context-free grammar (PCFG) (Foley et al., 1990) to control the random generation of unlimited shapes.

In a PCFG, a tree is randomly sampled given a set of probabilities. Starting from a root node, the leaf nodes of the tree keeps expanding according to a set of rules. The process is stopped until all 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 can be constructed by composing two other shapes through union and difference, and this construction can be recursively applied until all leaf nodes are a predefined set of concrete primitive shapes (terminals). The parameters can be 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:

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

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:

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

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

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

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

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

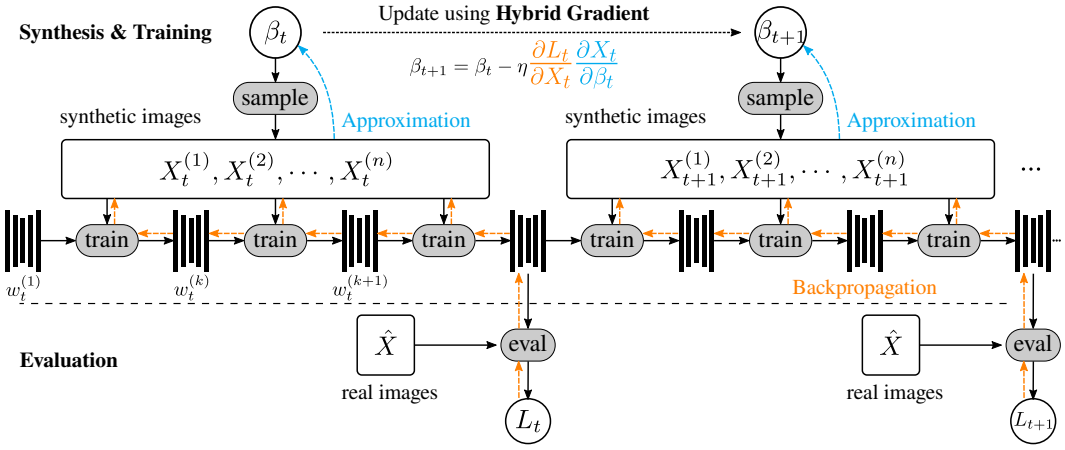

Figure 2: The details of using "hybrid gradient" to incrementally update β and train the network. The analytical gradient is computed by backpropagating through unrolled training steps (colored in orange). The numerical gradients are computed using finite difference approximation by sampling in a neighborhood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

4.2 Hybrid Gradient

After a deep network is trained on synthetic training data X, it 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 gradients: 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 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:

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

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 of the generalization loss L for each sample $X^{(k)}$ can be computed through backpropagation:

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

$$\frac{\partial L}{\partial w^{(k)}} = \frac{\partial L}{\partial w^{(k+1)}} \cdot \frac{\partial w^{(k+1)}}{\partial w^{(k)}} = \frac{\partial L}{\partial w^{(k+1)}} \cdot g_1'(w^{(k)}, X^{(k)}) \tag{8}$$

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

$$\frac{\partial L}{\partial w^{(n+1)}} = l_{\text{eval}}'(w^{(k+1)}, \hat{X}). \tag{9}$$

Aproximate 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, with Basic Random Search, a generalized finite difference method described in (Mania et al., 2018). First, we sample a set of noise from an uncorrelated multivariate Gaussian distribution (Mania et al., 2018):

$$\delta_1, \delta_2, \cdots, \delta_m \sim \mathcal{N}(\mathbf{0}, \sigma I) \tag{10}$$

Next, we approximate the Jacobian for each sample ($\otimes$ denotes cross product) (Mania et al., 2018):

$$\frac{\partial X^{(i)}}{\partial \beta} \approx \frac{1}{m} \sum_{j=1}^{m} \frac{f_{\mathcal{D}}(\beta + \delta_j, r_i) - f_{\mathcal{D}}(\beta - \delta_j, r_i)}{2\|\delta_j\|} \otimes \frac{\delta_j}{\|\delta_j\|} \tag{11}$$

Incremental training Following Yang & Deng (2018), we incrementally update parameters β and network weights w. At timestamp t, we update β_t with the hybrid gradient; for network weights, we simply use the latest trained network for initialization in timestamp $t+1$:

$$\beta_{t+1} = \beta_t - \gamma \frac{\partial L_t}{\partial \beta_t} = \beta_t - \gamma \sum_{i=1}^{n} \frac{\partial L_t}{\partial X_t^{(i)}} \cdot \frac{\partial X_t^{(i)}}{\partial \beta_t} \qquad w_{t+1}^{(1)} = w_t^{(n+1)} \tag{12}$$

5 EXPERIMENTS

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

Specifically, We experiment on the task of surface normal estimation on two real datasets: MIT-Berkeley Intrinsic Images Dataset (MBII) (Barron & Malik, 2015), which focuses on images of single objects and NYU Depth (Silberman et al., 2012), which focuses on indoor scenes. For the third dataset, we experiment on the task of depth estimation on the renderings of the scanned human faces in the Basel Face Model dataset (Paysan et al., 2009).

In all of our experiments, our networks are trained on synthetic images only, and the generalization loss is computed on the validation split of the datasets mentioned above. For MBII, we use pure synthetic shapes (Yang & Deng, 2018) 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 (Song et al., 2017) and augment the original 3D configurations in Zhang et al. (2017). For Basel Face Model, we sample synthetic faces from a morphable model and evaluate on renderings of scanned faces.

5.1 MIT-BERKELEY INTRINSIC IMAGES

Following the work of Yang & Deng (2018), we recover the surface normals of an object from a single image.

Synthetic shape generation In Yang & Deng (2018), a population of primitive shapes such as cylinders, spheres and cubes are evolved and rendered to train deep networks. The evolution operators are defined as transformations of individual shapes, as well as boolean operations of shapes in Constructive Solid Geometry (CSG) (Foley et al., 1990). 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_translate * 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 simply compute the mean angle error of predictions on the training set of the MIT-Berkeley dataset.

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
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

We compose our shape in mesh representations, slightly different from the implicit functions in Yang & Deng (2018). Therefore, we re-implemented their algorithm with mesh representations for fair comparison. For network training and evaluation, we follow Yang & Deng (2018) and train the Stacked Hourglass Network (Newell et al., 2016) 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 ($\leq N°$). See Appendix for detailed definition.

Ablation study We first sample 10 random values of β in advance, then for each β we train a network, with the exact same training and evaluation configurations as in our hybrid gradient. 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 initialize β_0 from these three values 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. This simple diagnostic experiment demonstrates that our algorithm is working properly.

Comparison with previous work In this experiment, we compare with black-box algorithms including Basic Random Search (Mania et al., 2018) and Shape Evolution (Yang & Deng, 2018). Because we use mesh implementation instead of implicit computation graph in Yang & Deng (2018) for CSG, we re-implemented Shape Evolution with the same setting for fair comparison. We follow Yang & Deng (2018) for the initialization of β, train the networks and update β for the same number of steps. We then report the test performance of the network which 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, and plot the test mean angle error with respect to the CPU time, GPU time and total computation time (Fig. 4). We see that our algorithm is more efficient than the above baselines. Shapes sampled from our optimized PCFG are shown in Fig. 3.

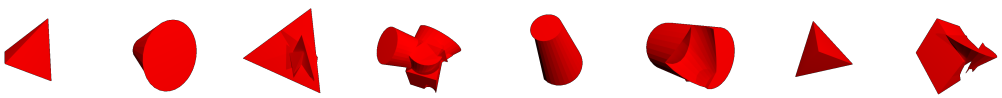

Figure 3: Sampled shapes from our probabilistic context-free grammar, with parameters optimized using hybrid gradient.

Table 2: Our approach compared to previous work, on the test set of MIT-Berkeley images (Barron & Malik, 2015). The results show that our approach is better than the state of the art as reported in Yang & Deng (2018).

	Summary Stats ↑			Errors ↓		
	$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	MAE	Median	MSE
SIRFS (Barron & Malik, 2015)	20.4%	53.3%	70.9%	26.2°	—	0.2964
Evolution (Yang & Deng, 2018)(Reported)	21.6%	55.5%	73.5%	23.3°	—	0.2204
Evolution (Yang & Deng, 2018)(Our Impl.)	23.0%	58.3%	73.8%	22.5°	**18.8°**	0.2042
Basic Random Search (Mania et al., 2018)	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**

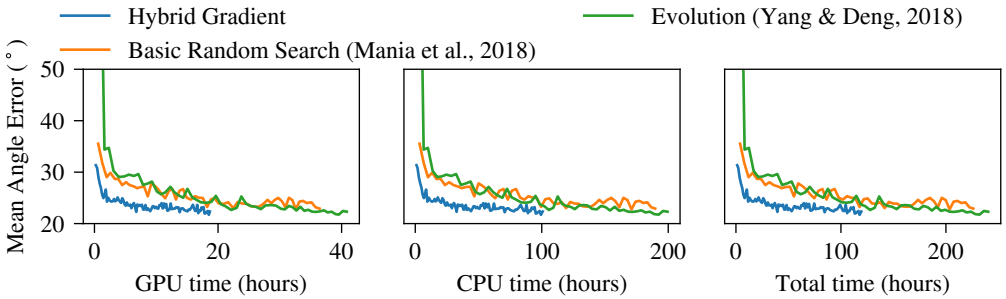

Figure 4: Mean angle error on the test images *vs*. computation time, compared to two black-box optimization baselines.

5.2 NYU Depth

Scene perturbation We design our scene generation grammar as an augmentation of collected SUNCG scenes (Song et al., 2015) with the cameras from Zhang et al. (2017):

```
S => E,P;
E => T_shapes * R_shapes * E0; P => T_camera * R_camera * P0;
T_shapes => translate(rand_x, rand_y, rand_z);
R_shapes => rotate_euler(rand_yaw, rand_pitch, rand_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 (Silberman et al., 2012) with the same setup as in Zhang et al. (2017). 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 using the original model in Zhang et al. (2017) 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 (Silberman et al., 2012), following the same protocol. The results are reported in Table 3. Note that none of these networks has seen a single real image except for validation.

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.

Table 3: The performance of the finetuned networks on the test set of NYU Depth V2 Silberman et al. (2012), compared to the original network in Zhang et al. (2017). The networks are only trained 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.

	Summary Stats ↑			Errors ↓	
	$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	Mean	Median
Original (Zhang et al., 2017)	24.1%	49.7%	61.5%	28.8°	22.7°
Augmentation with random β	23.0%	48.8%	61.3%	29.2°	23.2°
Agumentation with Hybrid gradient	**27.3%**	**52.5%**	**63.8%**	**28.1°**	**21.1°**

5.3 Basel Face Model

Synthetic face generation We exploit an off-the-shelf 3DMM morphable face and expression model (Dai et al., 2017; Zhu et al., 2015; 2016) to generating human 3D models, with face and pose parameters randomly sampled from mixtures of Gaussians or von Mises. Because the number of paramaters for 3DMM is over 400, we only include the first 10 principal dimensions for geometry, texture and expression parameters in the decision vector β, and uniformly sample for the remaining dimensions.

Evaluation We evaluate on the renderings of the scanned human faces (Paysan et al., 2009). 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. 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. (2014) and also detailed in the Appendix.

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.

	Relative Difference		RMSE		
	Absolute	Squared	Linear	Log	Log (scale inv)
Training with random β	0.03718	9.701×10^{-3}	0.1395	0.1014	0.09717
Basic Random Search (Mania et al., 2018)	0.02330	1.728×10^{-3}	0.0581	0.0299	0.02700
Hybrid gradient	**0.02256**	$\mathbf{1.649 \times 10^{-1}}$	**0.0570**	**0.0293**	**0.02603**

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.

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 and depth 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.

References

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*, pp. 3981–3989, 2016.

Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012. ISSN 1532-4435.

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, pp. 2546–2554. Curran Associates Inc., 2011. ISBN 978-1-61839-599-3.

Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2019. URL `http://www.blender.org`.

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.

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.) (ed.), *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pp. 611–625. Springer-Verlag, October 2012.

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.

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.

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.

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

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.

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 (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2366–2374. Curran Associates, Inc., 2014.

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 (eds.), *Advances in Neural Information Processing Systems 31*, pp. 9516–9527. Curran Associates, Inc., 2018.

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. ISBN 0-201-12110-7.

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.

David Ha, Andrew Dai, and Quoc Le. Hypernetworks. In *ICLR*, 2017.

M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1978. ISBN 0201029553.

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.

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)*, pp. 1168–1174, Nov 2011.

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.

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.

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.

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.

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, pp. 440–448. AUAI Press, 2914. ISBN 978-0-9749039-1-0.

Ke Li and Jitendra Malik. Learning to optimize. In *ICLR*, 2017.

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

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

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 (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1800–1809. Curran Associates, Inc., 2018.

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.

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, September 2018. ISSN 0920-5691.

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.

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

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*, pp. 483–499. Springer, 2016.

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 (eds.), *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2009, 2-4 September 2009, Genova, Italy*, pp. 296–301. IEEE Computer Society, 2009. doi: 10.1109/AVSS.2009.58.

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.

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.

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

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

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.

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.

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.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision – ECCV 2012*, pp. 746–760. Springer Berlin Heidelberg, 2012.

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*, pp. 567–576, 2015.

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.

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.

Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

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

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

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.

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.

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.

Dawei Yang and Jia Deng. Shape from shading through shape evolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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. ISSN 0730-0301.

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.

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*, pp. 787–796. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298679.

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*, pp. 146–155. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.23.

A APPENDIX

A.1 METRICS

Here we detail the metrics that we used in the paper. Assume $\mathbf{n}_i$ and $\mathbf{n}_i^*$ are the unit normal vector at i-th pixel (of N total) in the prediction and ground truth normal maps, respectively. d_i and d_i^* are depth values of the i-th pixel in the prediction and ground truth depth maps, respectively.

- Mean Angle Error (MAE): $\frac{1}{N}\sum_i \arccos(\mathbf{n}_i \cdot \mathbf{n}_i^*)$
- Median Angle Error (MAE): $\underset{i}{\text{median}}[\arccos(\mathbf{n}_i \cdot \mathbf{n}_i^*)]$
- Threshold δ: Percentage of $\mathbf{n}_i$ such that $\arccos(\mathbf{n}_i \cdot \mathbf{n}_i^*) \leq \delta$
- Mean Squared Error (MSE): $\frac{1}{N}\sum_i [\arccos(\mathbf{n}_i \cdot \mathbf{n}_i^*)]^2$
- Absolute Relative Difference: $\frac{1}{N}\sum_i |d_i - d_i^*|/d_i^*$
- Squared Relative Difference: $\frac{1}{N}\sum_i (d_i - d_i^*)^2/d_i^*$
- RMSE (linear): $\sqrt{\frac{1}{N}\sum_i (d_i - d_i^*)^2}$
- RMSE (log): $\sqrt{\frac{1}{N}\sum_i (\log d_i - \log d_i^*)^2}$
- RMSE (log, scale-invariant): $\sqrt{\frac{1}{N}\sum_i (\log d_i - \log d_i^* \cdot [\frac{1}{N}\sum_i (\log d_i - \log d_i^*)])^2}$

A.2 MIT-BERKELEY INTRINSIC IMAGE DATASET

Our decision vector β for PCFG is a 29-d vector, with 4 dimensions representing the probabilities of sampling different primitives, 2 for sampling union or difference, 1 for whether to expand the tree node or replace it with a terminal, 6 for translation mean/variance, 6 for scaling log mean/variance, 2 for sphere radius log mean/variance, 2 for box length mean and variance, 4 for cylinder radius and height log mean/variance, and 2 for tetrahedron length log mean/variance.

For optimizing β, we use the mean angle error loss on the validation set as the generalization loss, and use RMSprop (Tieleman & Hinton, 2012) to obtain the gradient for β. Note that some dimensions of β are constrained (such as probability needs to be non-negative), so we simply clip the value of β to valid ranges when sampling near β for finite difference computation and updating β. We present the qualitative results in Fig. 5.

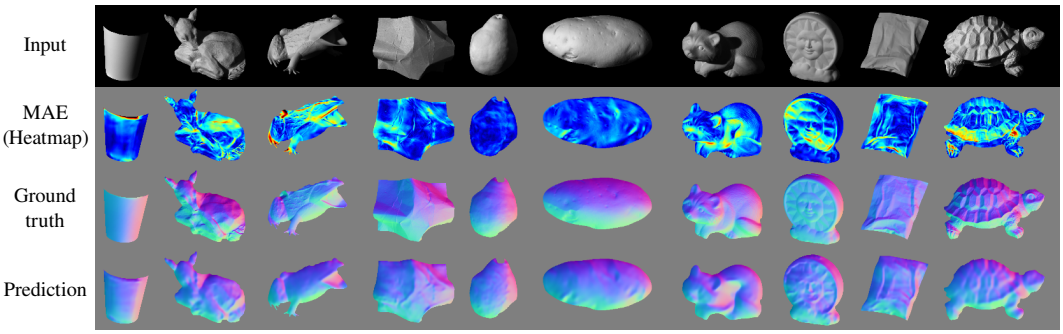

Figure 5: The test set of the MIT-Berkeley Intrinsic Images dataset.

A.3 NYU DEPTH V2

The decision vector β is 108-d. It includes the parameters for mixtures of Gaussians/Von Mises for 6 degree-of-freedom (vertical, horizontal and fordinal displacement, yaw, pitch, roll rotation) for shapes in the scene and the camera. Each mixture contains 9 parameters (3 probabilities, 3 means and 3 variances). Examples of perturbed scenes and the original scenes are shown in Fig. 6.

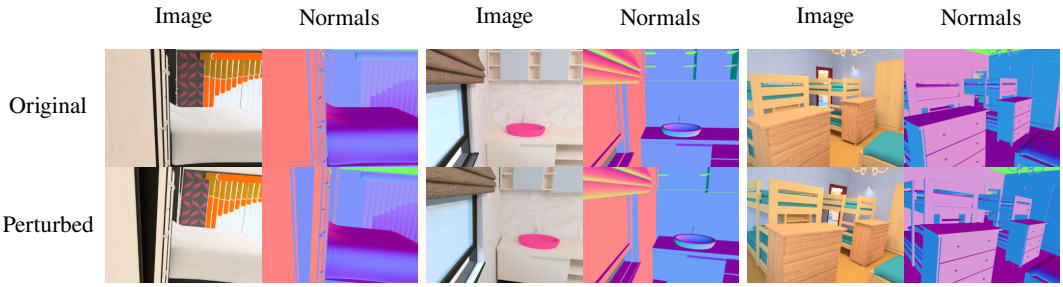

Figure 6: The original scenes in the SUNCG dataset, and our scenes with camera and objects perturbed using our PCFG.

A.4 BASEL FACE MODEL

The decision vector β has 204 dimensions. We use an off-the-shelf 3DMM implementation[1] to generate face meshes and textures for training. The 3DMM has 199 parameters for face identity, 29 for expression and 199 for texture. In implementation, we use 10 principal dimension for face/expression/texture respectively, and randomly sample from a mixture of 3 multivariate Gaussians. Note that the dimensions are independent, so we have a total of 183 parameters for generating the face mesh. For the 3-dof face pose angle, we also use mixtures of 3 von Mises, which have 21 parameters in total.

For rendering the training set, we apply human skin subsurface model using Blender(Blender Online Community, 2019), with a random white directional light uniformly distributed on $-z$ hemisphere. For rendering the test set (the scanned faces in the Basel Face Model), we render with the same 3 lighting angles and 9 pose angles, and the same camera intrinsics as in the original dataset. Fig. 7 shows The training images randomly generated by the PCFG (left) and example test images(right).

Synthetic faces generated from the PCFG

Re-renderings of real scans

Figure 7: Training images generated using PCFG with 3DMM face model, and 6 example images from the test set.

[1] https://github.com/YadiraF/face3d