Received: 8 November 2021 | Revised: 18 January 2022 | Accepted: 7 March 2022

IET Image Processing

DOI: 10.1049/ipr2.12486

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Image-level dataset synthesis with an end-to-end trainable framework

Zhenfeng Xue[1,2] | **Weijie Mao**[2] | **Yong Liu**[1,2]

[1]Research Center for Intelligent Perception and Control, Huzhou Institute of Zhejiang University, Huzhou, China

[2]Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China

Correspondence
Zhenfeng Xue, Research Center for Intelligent Perception and Control, Huzhou Institute of Zhejiang University, No.819 Xisaishan Road, Huzhou, China.
Email: zfxue0903@zju.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61633019

Abstract

Dataset synthesis via virtual engines like Unity is attracting much more attention in recent years due to its low cost at obtaining ground-truth labels. For this kind of work, virtual environments are constructed within the engine to mimic the real-world, either with great manual efforts or learning-based methods. The latter shows superiority over the former when the target real-world scenes are changeable, from which the attributes of environments can be automatically adjusted based on the distribution difference between the synthetic and real-world datasets. However, the non-differentiability of whole pipeline hinders the efficiency of attribute optimization. To this end, this paper proposes to simulate synthetic datasets from a fine-grained perspective, such that the system can be trained at an end-to-end manner. Specifically, it is converted into an image-level data synthesis problem, and designs a constraint using the content loss between two images. As the rendering process of virtual engine is mathematically unknown, which blocks the back propagation of the gradients, a generative model is trained to approximate the engine. As a result, the whole framework becomes fully differentiable and the attributes can be optimized efficiently by gradient descent. Experimental result shows the efficiency of our method in obtaining useful synthetic training datasets. Besides, it is found that the image-level method enables to learn the potential distribution of real-world data, which is hard to be achieved by existing methods. As far as we know, it is the first attempt to finish this task with a differentiable process.

1 | INTRODUCTION

Before we develop a computer vision application in a new scenario, it comes as a common practice to start from collecting and annotating real-world images [1–4]. Making a new dataset consumes much time and manpower, which may delay the progress of algorithm design. An alternative is to simulate synthetic datasets [5–7] via virtual engines like Unity. The potential advantage of this method lies in that the ground truth labels can be easily extracted from the engine. What is more, the parameterization process of scene construction makes it possible to control everything in the virtual environment, and changing the content of virtual scenes is likely to reduce the distribution difference between the simulated and real-world data. Unfortunately, it remains an open problem in modifying the virtual scene structures efficiently.

To narrow the content gap between the synthetic and real-world data, great manual efforts should be taken to change the appearance of virtual scenes. Recent works [8–11] review that the scene re-construction process can be implemented automatically under a learning-based framework. These methods optimize the attributes of virtual scenes, such as car rotation, pedestrian density and etc, under a training objective to minimize the distribution difference among the synthetic and real-world data. They usually use dataset-level metrics to make the constraint, like MMD [12], FID [13] or task accuracy. This prevents the whole pipeline to be differentiable mainly due to the following two reasons. First, the MMD or FID metric computes one score upon all the images in two datasets, and the mathematical equation of the gradients is too long to be expressed. Second, the rendering process of the virtual engine is mathematically unknown, which blocks the back propagation of

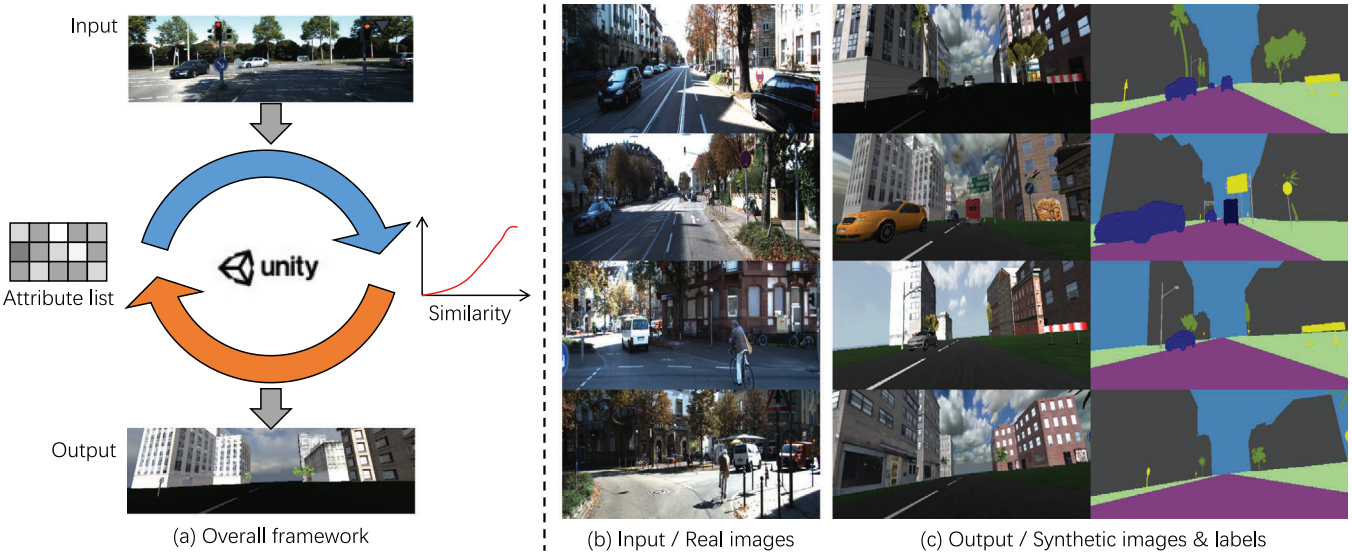


FIGURE 1 (a) shows the overall framework. Given one real-world image, our method is able to automatically adjust the attributes of the engine (Unity) to generate one synthetic image with high content similarity. (b) shows some example input images, and (c) shows the corresponding simulated images with ground truth labels acquired from the engine

the gradients. As a result, only REINFORCE [14] or brute force search algorithms could be employed to optimize the attributes. This affects the optimization efficiency to a large extent.

In order to tackle the system's being non-differentiable and improve the attribute optimization efficiency, this paper proposes the image-level data synthesis method from a fine-grained perspective. Specifically, we consider the standard scene reconstruction process as an attribute searching and image content similarity measurement process. The overall framework as well as some example images are shown in Figure 1. During each simulation iteration, only one image is generated according to the attributes, and we use the content loss in style transfer problem [15–17] as the constraint. Firstly, a parameterized dataset simulator is constructed within the virtual engine, where the scene structure can be modified by some external attributes. Then a set of attributes are randomly initialized and sent into the engine to generate a synthetic image. The simulated image is compared with a given real-world image to calculate a similarity score. Our method is able to automatically adjust the attribute values according to the score so that the simulated image shares a high appearance similarity with the real-world image.

Thanks to the image-level form, the gradients of the loss function are able to be calculated and back-propagated. In order to further alleviate the problem that the rendering process of the engine is unknown, we train a generative model to imitate the output of the engine according to the input, so as to approximate the gradients. As a result, the whole system becomes fully differentiable and the attributes can be optimized efficiently by gradient descent.

The differences between existing methods and ours are summarized in Table 1. All existing methods including LTS [8], Meta-Sim [9], Attr. Des. [10] and SDR [11] are based on a so-called dataset-level metric, where they use MMD, FID or task accuracy to compare the content similarity on a dataset level. Such a practice not only makes the whole learning system hard to be differentiable, but also costs more time during the simulation process. Because they need to generate a batch of synthetic images to calculate the distribution difference in each iteration. In contrast, our method simulate one synthetic image each time, so we call it the image-level dataset synthesis method relatively. The dataset-level methods mainly use reinforcement learning or brute force search to optimize the attributes. Such an approach lacks efficiency in a continuous search space, and the compromise is to discretize the attributes. While for our method, the whole training pipeline is fully differentiable so that gradient descent can be used to optimize the attributes. It maintains a high efficiency in even a continuous search space. Such a method ensures efficiency even in continuous search space. As far as we know, it is the first attempt to finish this task with a differentiable process.

Like [9], we validate our approach on the MNIST toy simulators in controlled settings, where our method is shown to bridge the distribution gaps efficiently. Especially when the target distribution is complex, existing dataset-level methods are hard to fit the distribution and the test performance drops rapidly. Our image-level method approximates each sample in the target distribution, so that it is able to potentially learn the distribution of real-world datasets. As a result, our method keeps a high test performance even in hard situations. The corresponding visualization results also verify that our method is able to fit the target distribution.

We further showcase our method on matching a real self-driving dataset, leading to improved content generation quality, as measured by synthetic-to-real performance. We modify the SceneX [11] engine to meet the image-level simulation process, and show performance gain after employing our method

TABLE 1 Differences between existing methods and ours. RL, FD, BF and GD denote the reinforcement learning, finite difference, brute force and gradient descent, respectively

Method	Constraint	Metric	Search space	Algorithm	Differentiable
LTS [8]	Dataset	Acc.	Continuous	RL	No
Meta-Sim [9]	Dataset	MMD & Acc.	Continuous	RL & FD	No
Attr. Des. [10]	Dataset	FID	Discrete	BF	No
SDR [11]	Dataset	Acc.	Discrete	RL	No
Ours	Image	Content loss	Continuous	GD	Yes

to optimize the attributes of the engine compared with random scenes.

In summary, the contributions are as follow.

- We formulate the image-level dataset synthesis framework that switches the scene re-construction process as an attribute searching and image similarity measurement problem.
- We complete the dataset simulation problem with a fully differentiable process, so that gradient descent can be used to maintain the efficiency of attribute optimization.
- We validate the effectiveness of our method through experiments, and find it is able to learn the potential distribution of target dataset, which is hard to be achieved by existing methods.

2 | RELATED WORK

2.1 | GAN-based data generation

GAN-based data generation methods [18–20] aims to adjust the style of synthetic images to that of real-world images. For this kind of work, image-level domain adaptation [20] via adversarial training has proven to be an effective way in several works such as [21–23]. To generate data that has similar appearance with the target data, Zhang et al. [24] use an appearance adaptation network to gradually generate a stylized image from a noise image by adapting the appearance to target data. Chen et al. [25] further propose an input-level adaptation network that leverages the depth information to reconstruct the source image. It employs an adversarial learning [26] framework to ensure style consistency between source and target domains.

2.2 | Dataset-level synthetic data simulation

In contrast to previous works [5–7, 27–30] that design realistic worlds within virtual environments with great manual efforts, recent researches begin to focus on learning-based synthetic content simulation. Among them, Ruiz et al. [8] propose to use the test accuracy of a main task model as the reward to optimize the distribution parameters with REINFORCE [14] algorithm. Kar et al. [9] further propose to use a graph model to represent the scene structures and employ Graph Convolutional Networks (GCNs) [31] to optimize the attributes with MMD [12] metric and task accuracy. Yao et al. propose to use FID [13] with attribute descent to simulate Vehicle ReID dataset. Due to the optimization inefficiency of the above methods in dealing with complex scenes, Xue et al. [11] define 23 global attributes over a driving scene and propose the SDR in groups with task accuracy as the reward to optimize the scenes. Overall, the current methods perform distribution matching over the statistical information with a dataset-level metric. Recently, Kar et al. [32] propose to calculate the KL divergence over the samples as the reward, and it gives us inspiration towards image-level data simulation.

2.3 | Differentiable rendering engine

Content simulation via virtual engines suffers from the non-differentiability of the whole pipeline due to the complexity of the rendering process. By far, there are two mainstream ways to achieve differentiable renderer. On the one hand, soft rasterizer [33, 34] has become a mature technology in computer graphics. It simulates the rendering process through 3D meshes, and blurs the edges so that the gradients of rasterization can be calculated. Genova et al. [35] apply the differentiable rasterizer into a 3D face generation model, so that the whole pipeline can be trained at an end-to-end manner. On the other hand, physics-based rendering has become a hot topic. Li et al. [36] propose an edge sampling method to calculate the derivatives of visibility. Loubet et al. [37] further propose to replace edge sampling with reparameterization to speed up the process. Following works [38, 39] further extend the methods to volume and path space rendering. The above methods are hard to be applied to data simulation due to their complexity in implementation. Recently, Shi et al. [40] employ a convolutional neural network to imitate the behaviour of a face generation engine and our method is in line with them in tackling the non-differentiability.

2.4 | Neural style transfer

The neural style transfer (NST) problem [15–17] aims to generate one stylized image that has the same content with given one. The training objective is as follows.

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s, \tag{1}$$

where $\mathcal{L}_c$ and $\mathcal{L}_s$ denote content and style loss, and λ controls the balance of the above two constraints. The difficulty lies in the measurement of the image similarity between two images. Current NST methods can be divided into two parts, that is, the global methods [15, 41–43] and the local methods [44, 45], where the former measures the style similarity based on the statistical analysis of global features, and the latter performs the patch-level matching to better preserve the local details. The hybrid method [46] is proposed more recently to combine both advantages of global and local methods . However, these methods are specific designed for image-to-image translation rather than a data simulation method thus can not be directly applied to virtual environments.

3 | METHOD

In this section, we first formulate the image-level data synthesis problem and introduce the corresponding training pipeline. Then we explain how to solve the system's being non-differentiable, through replacing the dataset-level metric with the content loss among two images and training a generative model to approximate the virtual engine. Finally, we discuss about the potential ability of our method in fitting the target distribution.

3.1 | Problem statement

Given a group of target images X_t, that are captured from the real-world, the purpose of dataset synthesis is to simulate a group of synthetic images X_s with ground truth labels Y_s via virtual engines like Unity or GTA V. Traditional methods [5–7] focus on designing realistic environments with great manual efforts. Recent researches [8–11] begin to simulate the images with learning-based methods. The training objective is to minimize the distribution difference between the synthetic and real-world data under a dataset-level constraint,

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\xi}\sim\mathbf{P}_{\boldsymbol{\phi}}(\boldsymbol{\xi})} \boldsymbol{D}(\mathbf{X}_s, \mathbf{X}_t), \tag{2}$$

where $\boldsymbol{\phi}$ denotes the distribution parameters to be learned. $\boldsymbol{\xi}$ contains the attributes sampled from distribution P, and $\boldsymbol{D}$ is a distribution metric to measure the difference between $\mathbf{X}_s$ and $\mathbf{X}_t$, usually using MMD [12], KID [47], FID [13] or task accuracy [11].

Differently, we formulate the dataset synthesis problem as an attribute searching and image content similarity measurement problem from an image-level perspective. Given a target image $\mathbf{x}_t$, the objective is to directly learn a set of attributes $\boldsymbol{\xi}$ for the virtual engine, so that a synthetic image $\mathbf{x}_s$ can be generated with similar content. The training objective switches to:

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} \mathcal{L}_c(\mathbf{x}_s, \mathbf{x}_t), \tag{3}$$

where $\mathcal{L}_c$ denotes the content loss function, and $\boldsymbol{\xi}$ denotes the attributes to be learned corresponding to $\mathbf{x}_s$. Such a modification has the following advantages again the dataset-level framework.

First, it is time-saving for the training process. For each iteration, the dataset-level methods need to generate a group of images, so that the distribution difference is enough to be measured. There exists much repetitive generation work, while in our case only one image is generated. Second, the image-level pipeline makes it possible to use gradient descent to optimize the attributes. For dataset-level methods, the derivative of MMD or FID is hard to be calculated, while this is easy to be achieved through using the content loss. This makes it easier to optimize the correlated attributes together, as Xue et al. [11] have emphasized the importance of joint optimization for the learning process. Third, instead of pre-defining the distribution function like the dataset-level methods, we are intended to learn the potential distribution. This is significant when the target distribution is unknown for a new scene.

3.2 | Image-level training pipeline

The training pipeline of image-level data synthesis is illustrated in Figure 2. There are three main parts, including a virtual engine $\mathbf{R}$, a generative model $\mathbf{G}$ and a feature extractor $\mathbf{F}$. Virtual environment is manually designed within the engine $\mathbf{R}$, and its structure is fully controlled by the pre-defined external attributes of the engine, such as rotation degree, translation distance, etc. The engine takes a set of attributes $\boldsymbol{\xi}$ as input, and outputs a rendered image $\mathbf{x}_s$:

$$\mathbf{x}_s, \mathbf{y}_s = R(\boldsymbol{\xi}), \tag{4}$$

where the corresponding label $\mathbf{y}_s$ is automatically generated through the rendering buffer. By modifying the attributes $\boldsymbol{\xi}$, the simulated image as well as the label can be updated accordingly.

Given a real-world image $\mathbf{x}_t$ that is to be mimicked, the attributes $\boldsymbol{\xi}$ are initially randomized and a synthetic image $\mathbf{x}_s$ is simulated. Then $\mathbf{x}_s$ and $\mathbf{x}_t$ are sent into the feature extractor $\mathbf{F}$ to extract content features. The feature extractor is composed of a neural network, such as VGG [48] or others. Then, the content difference between the two images can be measured by the following equation.

$$\begin{aligned} \mathcal{L}_c(\boldsymbol{\xi}) &= \mathcal{L}_c(\mathbf{F}(\mathbf{x}_s), \mathbf{F}(\mathbf{x}_t)) \\ &= \mathcal{L}_c(\mathbf{F}(\mathbf{R}(\boldsymbol{\xi})), \mathbf{F}(\mathbf{x}_t)). \end{aligned} \tag{5}$$

Except for the virtual engine, all of the above gradients can be calculated and back-propagated. Fortunately, we find that during the attribute optimization process, there is no need to generate the ground truth label. That is to say, we are able to train a generative model $\mathbf{G}$ to approximate the output of engine. The generative model is also a neural network, taking the attributes $\boldsymbol{\xi}$ as input and outputting a simulated image.

$$\mathbf{G}(\boldsymbol{\xi}) \approx \mathbf{R}(\boldsymbol{\xi}) = \mathbf{x}_s. \tag{6}$$

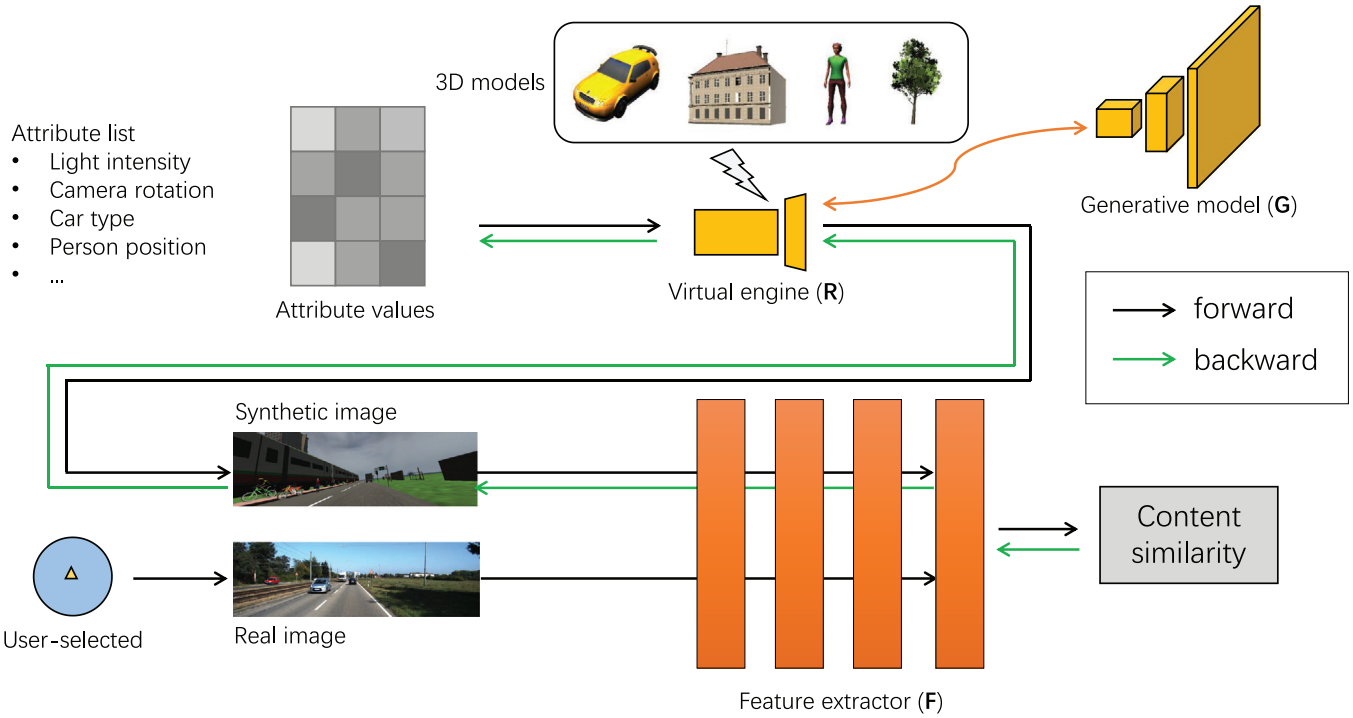


FIGURE 2 The whole framework of the proposed image-level dataset synthesis method. Inputting a set of randomly initialized attributes into the virtual engine, a synthetic image can be generated, which is then fed into a feature extractor along with a given real-world image. Then image content similarity between the two images is measured and the corresponding gradients are calculated and back-propagated to the input attributes, so that the attributes can be updated

By doing this, it enables to approximate the gradients of the engine during back-propagation. Specifically, during the attribute training steps, we replace **R** with **G**. Although this may bring noises to the gradients, we find it enough to get good results. Thus, the attributes can be optimized by gradient descent.

$$\begin{aligned}\boldsymbol{\xi}' &= \boldsymbol{\xi} - \eta \cdot \nabla_{\boldsymbol{\xi}} \mathcal{L}_c(\mathbf{F}(\boldsymbol{\xi})), \\ &\approx \boldsymbol{\xi} - \eta \cdot \nabla_{\boldsymbol{\xi}} \mathcal{L}_c(\mathbf{F}(\mathbf{G}(\boldsymbol{\xi})), \mathbf{F}(\mathbf{x}_t)),\end{aligned} \tag{7}$$

where η denotes the learning rate. For each real-world image $\mathbf{x}_t$, starting from a set of initialized attributes $\boldsymbol{\xi}$,we are able to obtain a best set of attributes $\boldsymbol{\xi}^*$ after achieving convergence of Equation 7. The algorithm for attribute training is presented in Algorithm 1.

After training, we obtain a set of optimized attribute values $\boldsymbol{\xi}^*$ for each input target image. During the testing stages, we input the attributes $\boldsymbol{\xi}^*$ into the virtual engine R to simulate a synthetic image and its label. For each target image, we are able to simulate a synthetic image-label pair, where the simulated image has small content difference with the target one.

ALGORITHM 1 Training phase of image-level dataset synthesis

Given:	$\mathbf{R}, \mathbf{G}, \mathbf{F}, \mathbf{X}_t, \eta$.
Hyperparameters:	I, N. ⟶ number of iterations and images
Pre-training:	**G**, **F**. ⟶ use samples from **R**
To be solved:	$\mathbf{X}_s, \mathbf{Y}_s$.
1:	**while** n ≤ N **do**
2:	initialize attributes ξ;
3:	**while** i ≤ I **do**
4:	$\mathbf{x}_s = \mathbf{G}(\xi)$; ⟶ simulate a synthetic image with **G**
5:	$\mathcal{L} = \mathcal{L}_c(\mathbf{F}(\mathbf{x}_s), \mathbf{F}(\mathbf{x}_t))$; ⟶ calculate the content loss
6:	$gradient = \nabla_{\xi} \mathcal{L}_c$; ⟶ calculate the gradients
7:	ξ -= $\eta \cdot gradient$; ⟶ update the attributes with gradient descent
8:	**end while**
9:	$\mathbf{x}_s, \mathbf{y}_s = \mathbf{R}(\xi)$; ⟶ render an image and its label with the engine
10:	**end while**

3.3 | Content similarity measurement

Like the neural style transfer problem [15–17], we employ an image content loss to measure the content similarity between the synthetic and real-world image. We choose the cosine distance of the feature extractor output as the content loss.

$$\begin{aligned}\mathcal{L}_c(\boldsymbol{\xi}) &= 1 - \cos(\mathbf{F}(\mathbf{x}_s), \mathbf{F}(\mathbf{x}_t)) \\ &= 1 - \cos(\mathbf{F}(\mathbf{R}(\boldsymbol{\xi})), \mathbf{F}(\mathbf{x}_t)) \\ &\approx 1 - \cos(\mathbf{F}(\mathbf{G}(\boldsymbol{\xi})), \mathbf{F}(\mathbf{x}_t)),\end{aligned} \tag{8}$$

where the cosine distance between two vectors $\mathbf{v}_s$ and $\mathbf{v}_t$ is expressed as follows.

$$\cos(\mathbf{v}_s, \mathbf{v}_t) = \frac{\langle \mathbf{v}_s, \mathbf{v}_t \rangle}{|\mathbf{v}_s| \cdot |\mathbf{v}_t|}. \tag{9}$$

Unlike [16] that uses the layer3's features of VGG16 [48] as the measurement criteria, we find it better to use the features of the task model. For example, if we are intended to simulate a synthetic dataset for semantic segmentation, and DeepLabv3 [49] is selected as the task model. Then it is better to use the output of DeepLabv3 to calculate the image content loss during attribute training.

3.4 | Approximation of virtual engine

We train a generative model to approximate the virtual engine, so that the simulation pipeline becomes fully differentiable. Like [40], we take a similar form of DAGAN [50] as the generative model, and formulate it as a regression problem. The training process of the generative model is to minimize the difference between $\mathbf{G}(\boldsymbol{\xi})$ and $\mathbf{R}(\boldsymbol{\xi})$ at raw pixels.

$$\mathcal{L}_{\mathbf{G}}(\boldsymbol{\theta}) = \mathbb{E}_{\xi \sim \mathbf{U}(\xi)} \|\mathbf{G}_{\boldsymbol{\theta}}(\boldsymbol{\xi}) - \mathbf{R}(\boldsymbol{\xi})\|_1, \tag{10}$$

where $\boldsymbol{\theta}$ denotes the parameters of $\mathbf{G}$ to be learned, and $\mathbf{U}$ denotes the distribution function of engine attributes $\boldsymbol{\xi}$. We use l_1 loss function rather than l_2 and l_1 encourages less blurring.

We use uniform distribution to randomly sample a large number of $\boldsymbol{\xi}$ and $\mathbf{R}(\boldsymbol{\xi})$ from the virtual engine so that the generative model can be trained on. After obtaining the generative model, we fix its parameters during attribute training.

3.5 | Discussion

(1) **What's the differences between the proposed method with GAN methods?**They have two main differences. Firstly, given a real-world image $\mathbf{x}_t$, GAN methods aim to simulate a synthetic image $\mathbf{x}_s$. While for the proposed method, it aims to simulate a synthetic image $\mathbf{x}_s$ and the label $\mathbf{y}_s$. The usage of virtual engine enables to extract the ground truth label easily, and the label can change with the image. Secondly, for GAN methods, they are able to change the style of images, such as light and color. While for the proposed method, it is targeted at changing the content of images, such as rotation degree and translation distance.

(2) **Is image-level simulation able to fit the target distribution?**Given a group of real-world images $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ that are sampled from an unknown distribution, we are able to obtain a group of synthetic images $\{\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_n\}$ through image-level data synthesis. Suppose the engine is designed to be good enough to mimic the real-world scenes, after attribute training, we can get:

$$\mathbf{x}'_1 \approx \mathbf{x}_1, \mathbf{x}'_2 \approx \mathbf{x}_2, \dots, \mathbf{x}'_n \approx \mathbf{x}_n. \tag{11}$$

Then the statistical information of the synthetic and real-world images is empirically equal. This is to say, we are able to fit the target distribution by mimicking each sample in the distribution.

(3) **Without MMD and FID, can the effectiveness of the proposed method be fully confirmed?**MMD and FID methods ensure the distribution similarity by narrowing the statistical indicators, such as mean and standard deviation, between the synthetic and real-world images. By comparison, our proposed method aims to solve this problem from a more fine-grained perspective. We are intended to simulate and approximate each sample in the real-world dataset, as such we are able to obtain a synthetic dataset that has small domain gap with the given one. By doing this, the statistical indicators can be ensured at the same time by the following expression.

$$\mathcal{D}(\mathbf{X}_s, \mathbf{X}_t) \to 0, \tag{12}$$

where $\mathcal{D}$ denotes the MMD or FID criterion.

4 | EXPERIMENT

In this section, we evaluate the proposed method on two different tasks. The following subsections follow a general structure where we first outline the desired task, the target data and task network. Then we introduce the associated virtual engine that generates labelled synthetic data, as well as the generative model used to approximate the engine. We first show qualitative results after training the generative model, and then show quantitative and qualitative results after training the task network using synthetic data generated by our method. We show strong boosts in quantitative performance and noticeable qualitative improvements in content-generation quality.

Similar to [9], the first experiment presented is in a controlled setting. The aim here is to probe the capability when the target distribution is considered unknown.

4.1 | MNIST

We first evaluate our approach on digit classification on MNIST-like data. The scenes are constructed with a background texture, one digit texture from the MNIST [51] dataset, and then samples a rotation and translation for the digit. Different from [9], the rotation and translation values are sampled from a multi-dimensional Gaussian.

Rotating MNIST. In this experiment, we manually generate random samples that are upright and centered, like regular MNIST digits as initial state. The target images $\mathbf{X}_t$ and test set $\mathbf{T}$ are images (with 48×48 resolution) where digits are centered and rotated (Figure 3 top row). The rotation degrees are sampled from a multi-dimensional Gaussian distribution. Ideally, the model will learn this exact transformation, and rotate the digits to the target degree in each image. We also show the

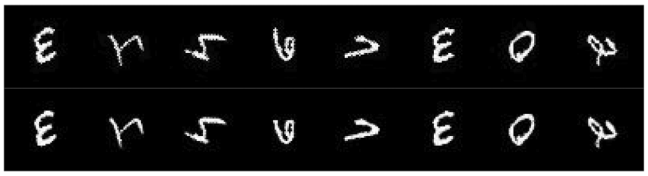


FIGURE 3 Examples from the rotated MNIST data. (top) The engine output, (bottom) the generative model output

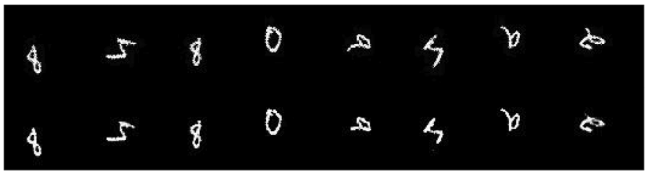


FIGURE 4 Examples from the rotated and translated MNIST data. (top) The engine output, (bottom) the generative model output

output of pre-trained generative model in this case (Figure 3 bottom row).

Rotating and translating MNIST. In this experiment, we additionally add translation to the distribution gap, making the task harder. We generate $\mathbf{X}_t$ and $\mathbf{T}$ (with 64×64 resolution) where in addition to being rotated, the digits are moved (Figure 4 top row). The target positions are also sampled from a multi-dimensional Gaussian distribution. We also show the output of pre-trained generative model in this case (Figure 4 bottom row).

Task network. Like [9], our task network is a small two-layer CNN followed by fully connected layers. We apply dropout in the fully connected layers (with 50, 100, and 10 features). We do not use data augmentation while training as it might interfere with our model's training by changing the configuration of the generated data, making the task optimization signal unreliable.

Virtual engine. The virtual engine here is implemented with Python. It transforms the texture based on the sampled transformation and pastes onto a canvas.

Generative model. The generative model consists of two fully connected layers and four transposed convolution layers. The fully connected layers map the input attributes to a latent vector, and then the transposed convolution layers increase the image resolution gradually. We use Adam optimizer for training with batch size of 64, with a learning rate of 0.002, and the training stops after 1000 epochs.

Quantitative results. Table 2 shows the training results when we use the synthesized dataset as training set and validate on target dataset. Here the target dataset varies in different cases. For example, in the rotation experiment, 'G5' denotes the rotation degree obey a five-dimensional Gaussian distribution and the same is true for others. Training directly on the initial scenes, that is, generating upright and centered digits in this case, results in just above random performance.

For rotating MNIST experiment, we add a multi-dimensional Gaussian distribution on the target rotations. With the increase of target distribution complexity, the test performance drops rapidly for existing methods, like Meta-Sim [9], attribute descent [10], and SDR [11]. The reason is that these methods narrow the distribution gap on the dataset-level metric by using mean and standard deviation, but are hard to match the potential distribution. While for our method, it keeps good performance for all cases. Because we are able to approximate each sample in the target dataset, and learn the potential distribution, showing the ability of our method to bridge the rotation gap.

For rotating-and-translating MNIST experiment, we further add a multi-dimensional Gaussian distribution on the target positions. Our method behaves inferior to existing methods when the target distribution is sample, like one-dimensional Gaussian, because there are many noisy examples in our result.

TABLE 2 Quantitative analysis of MNIST experiment. The target scenes are all sampled from multi-dimensional Gaussian distributions, denoted by 'Gx', where x=1,2,3,4,5

Transformation distribution	Rot.				Trans.			Rot. & Trans.		
	G1	G2	G3	G5	G1	G2	G4	G1	G2	G4
Initial scenes	10.9	11.2	14.2	17.8	16.9	13.1	11.4	14.7	12.1	10.7
Meta-Sim [9]	99.5	65.3	50.2	39.2	25.1	12.6	10.4	14.5	11.6	10.6
Attr. Des. [10]	99.9	71.3	49.7	39.6	99.0	55.8	34.7	98.8	**53.4**	31.1
SDR [11]	99.9	71.3	50.3	39.2	**99.4**	55.8	31.4	**98.9**	52.1	32.3
Ours	**99.9**	**99.8**	**99.8**	**99.8**	72.7	**75.9**	**72.1**	48.6	44.3	**39.5**

TABLE 3 Measurement of various scores for different methods

Distribution Score	Rot. G5			Trans. G4		
	MMD	FID	Acc.	MMD	FID	Acc.
Meta-Sim	4.8	32.7	39.2	28.1	76.8	10.4
Attr. Des.	7.9	26.1	39.6	18.1	32.9	34.7
SDR	8.2	26.5	39.2	18.7	36.3	31.4
Ours	**0.7**	**5.9**	**99.8**	**1.02**	**15.3**	**72.1**

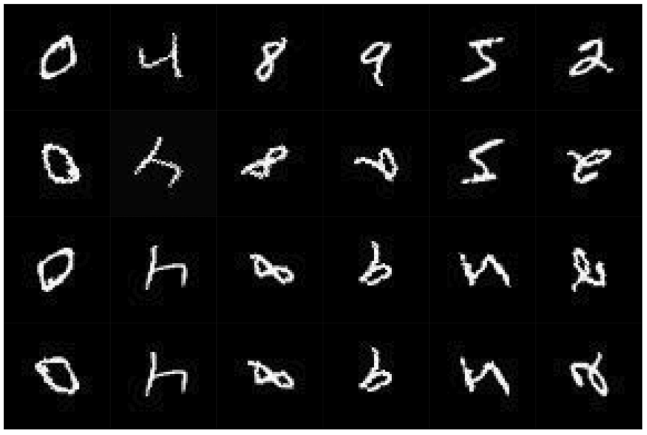


FIGURE 5 Example images for rotating MNIST data. (First row) Initial scenes. (Second row) Generated images by Meta-Sim. (Third row) Generated images by our method. (Fourth row) Target scenes with rotation gap with initial scenes

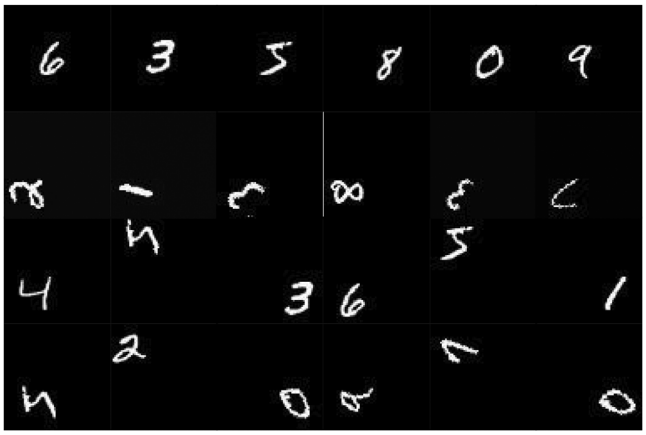


FIGURE 6 Example images for rotating-and-translating MNIST data. (First row) Initial scenes. (Second row) Generated images by Meta-Sim. (Third row) Generated images by our method. (Fourth row) Target scenes with rotation gap with initial scenes

But when the target distribution becomes complex enough, such as four-dimensional Gaussian, our method achieves a better performance. That's because of the potential ability of our method in fitting the target distributions.

Taking a closer look at the final results, we can find our method is also able to minimize the dataset-level metrics, such as MMD or FID. As shown in Table 3, our method obtains the highest test accuracy in 'Rot. G5' and 'Trans. G4' experiments. Meanwhile, we also obtain the lowest MMD and FID scores. This sufficiently verifies the effectiveness of the image-level dataset synthesis method.

Qualitative results. Taking a closer look at the simulated results. Figures 5 and 6 show the images for initial scenes, Meta-Sim output, our output and the target scenes for the two settings. Existing methods, like Meta-Sim, all try to narrow the distribution gap at a dataset-level, so they fail to fit each example in the target dataset. As a result, the rotation degrees are not always same with the target. While for our method, it simulates images at an image-level, so we exactly obtain a similar image with the target one, except that some examples have a rotation difference about 180 degrees. In Figure 6, although the simulated results are not so good for all methods, our method shows better performance at predicting the translation distance for the numbers.

Figure 7 visualizes the distribution degree distribution for each circumstance. The initial scenes samples rotation degree using Gaussian distribution, with 0 and 10 as mean and standard deviation. The target scenes contain a multi-dimensional Gaussian. In these cases, existing methods fail to predict the target distribution especially when the distribution function is complex, such as five-dimensional Gaussian distribution in the third row. While for our method, it is able to learn the target distribution.

Figure 8 visualizes the position distribution for each circumstance. In this experiment, existing methods fail to fit the distribution of target data, while our method still covers most of the cases. But our method brings much more noisy examples.

The reason for the noisy examples may be as follow. Here we also visualize the sensitivity of the content loss to the rotation gap and translation gap as shown in Fig. 9, and find that the rotation sensitivity is much better. That's may be because CNN is not sensitive to translation, and that's why the rotating result is better than translating result in our experiment. In the next version, we will try to fix this problem by designing more robust similarity criteria.

4.2 | Driving scenes

After validating our approach on controlled experiments in a simulated setting, we now evaluate our approach for **semantic segmentation** on the challenging KITTI dataset. We use the

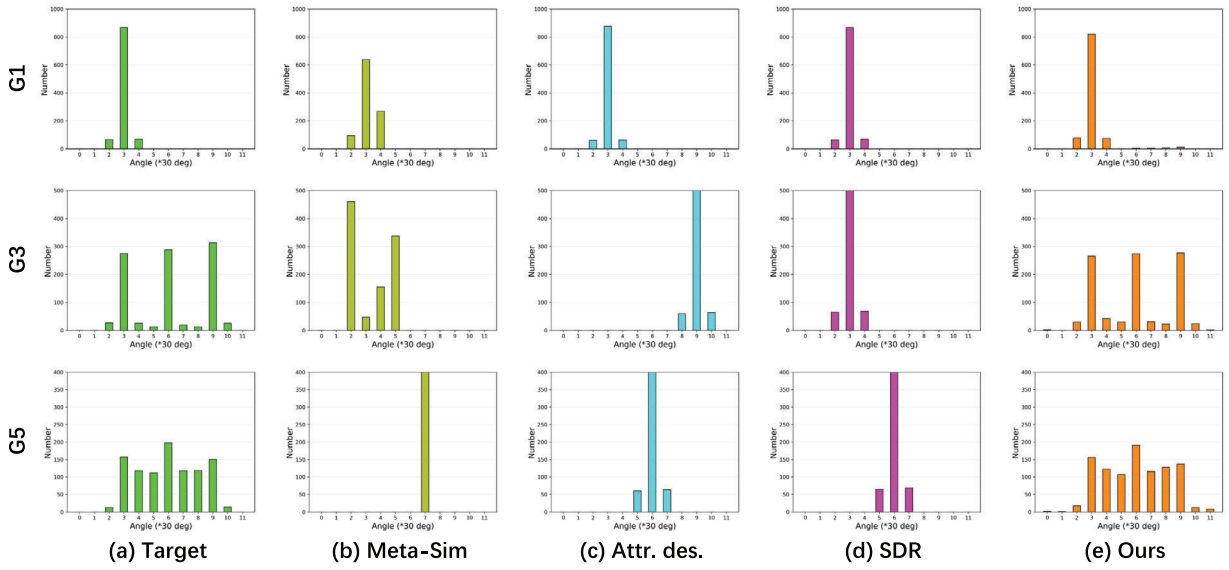


FIGURE 7 Visualization of the distribution of rotation degrees in initial scenes, Meta-Sim output, our output and target scenes for MNIST rotation gap

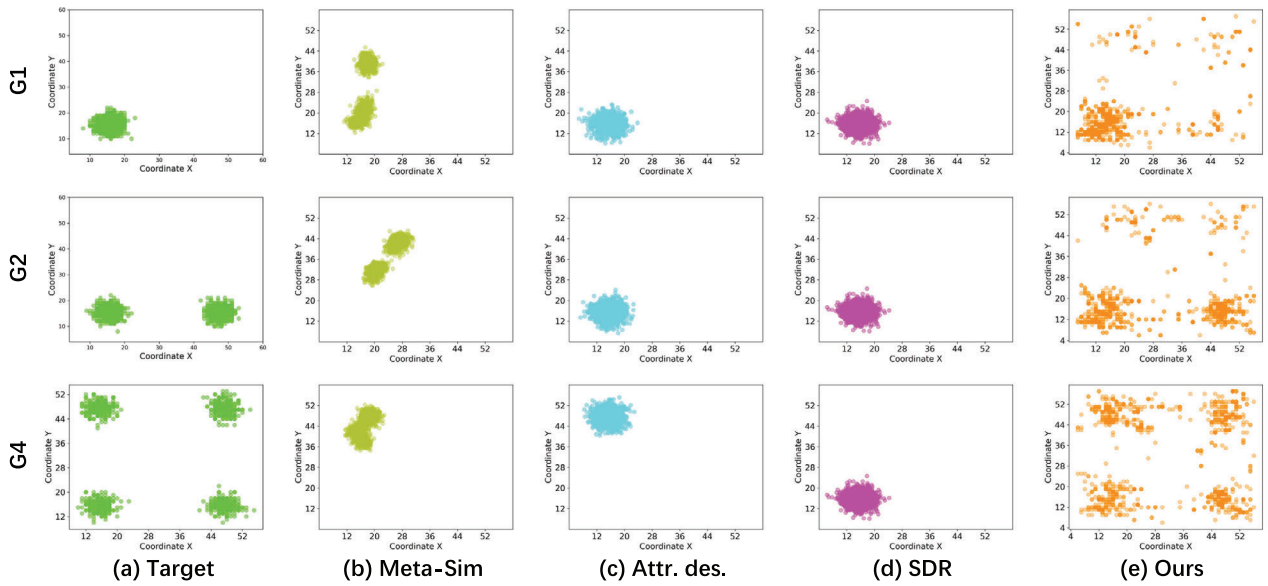


FIGURE 8 Visualization of the distribution of translation positions in initial scenes, Meta-Sim output, our output and target scenes for MNIST translation gap

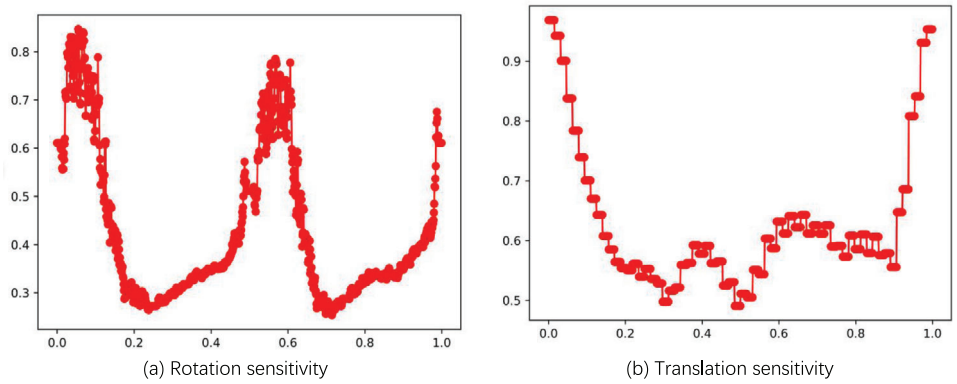


FIGURE 9 Sensitivity of the content loss to rotation gap and translation gap

SceneX engine from [11] as the simulator and make small modifications so that it can be used in our case.

Task network. We use DeepLabv3 [49] as our task network. It is also used as the feature extractor during attribute training. We use the final feature to judge pixel similarity of simulated images with real-world images.

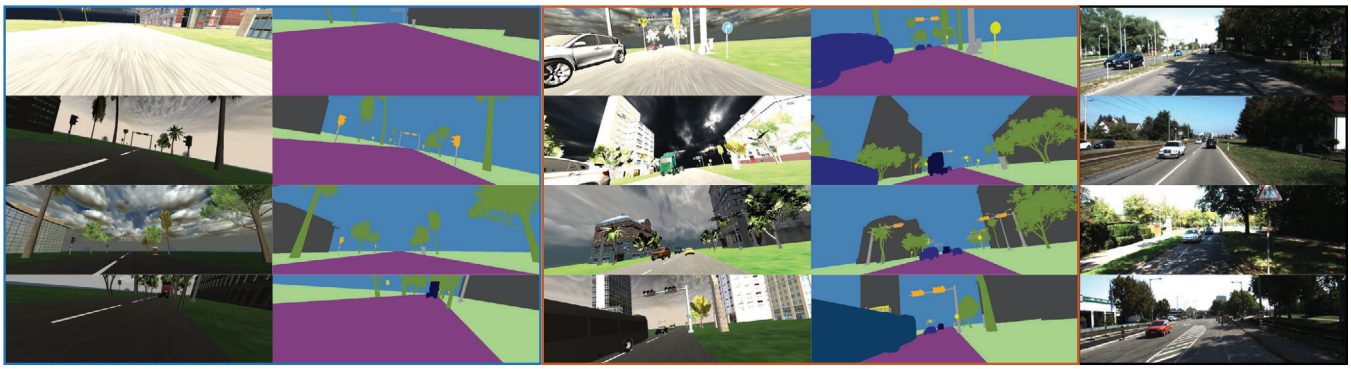
Generative model. The generative model consists of two fully connected layers and eight transposed convolution layers. We use Adam optimizer with a learning rate of 0.002, and train for 1000 epochs. The training results are shown in Figure 11.

Experimental setup. Following [11], we use scene segmentation as our task. Test data is formed by 200 images from KITTI semantic segmentation train set. We use the test set as the target images $\mathbf{X}_t$ since the semantic labels are not available. We report results on KITTI val set.

Quantitative results. Table. 4 reports the quantitative results of task network trained using data generated from initial scenes, learnt scenes by our method and Virtual-KITTI, when tested on KITTI. The learnt scenes exceed the initial scenes about 5.2% mIoU, showing its ability in bridging the distribution gap for our method. The Virtual KITTI dataset exceeds the learnt data by 7.8% mIoU. However, the Virtual KITTI is manually designed by a group of professional technicians with great efforts. While for our method, it is able to generate a dataset by machine learning. The generated quality is greatly influenced by the

TABLE 4 Quantitative results of synthetic to real KITTI dataset simulation

data	mIoU (%)
VKITTI	46.9
Initial scenes	33.9
Learnt scenes by Meta-Sim [9]	35.3
Learnt scenes by Ours	39.1



(a) Initial scene images & labels (b) Learnt scene images & labels (c) Target scene images

FIGURE 10 Visualization of the simulated images

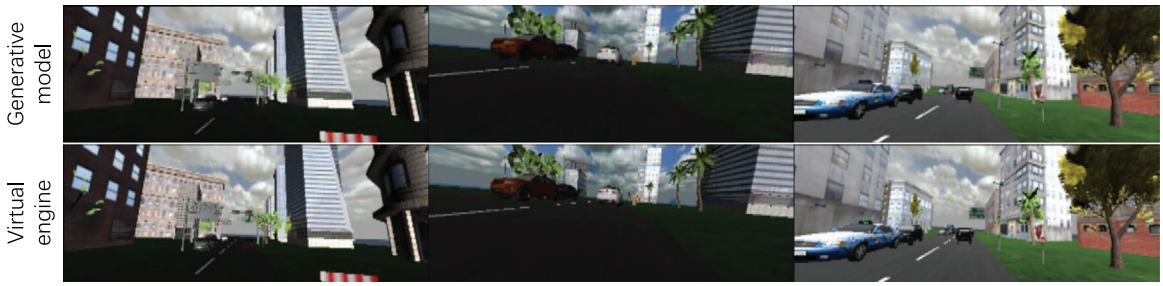


FIGURE 11 Output of the generative model and the engine

virtual engine. In the next version, we will design more realistic engines. Compared with Meta-Sim, our method obtains an about 3.8% performance again, showing its strong ability in simulating synthetic datasets compared with existing methods.

Qualitative results. Figure 10 shows the qualitative results for initial scenes and learnt scenes. The learnt scenes look much more similar than the initial scenes. For example, some cars and trees appear at the same location with the target scenes. In SceneX [11], the objects are all placed on different splines, which imposes a strong constraint in the scene structure. As such, the structure of the generated scene is too simple, and hard to be compared with manually designed scenes. That's why there still exists a large gap between the learnt scenes with the real-world ones.

5 | CONCLUSION

Annotating the real-world images is a time-consuming and labor-intensive job. In order to overcome this limitation for new-coming computer vision tasks, data synthesis via virtual engines has becoming a promising solution, since it is able to generate ground truth labels easily at a large scale. The difficulty lies in reducing the content gap between the real-world and virtual scenes. Previous dataset-level simulation methods focus on narrowing the distribution gap between the two datasets, using the REINFORCE algorithm to optimize the scene structures due to the non-differentiability of the engine. This paper proposes the image-level data synthesis pipeline that tackles the system's being non-differentiable, such that the attributes can be optimized at an end-to-end manner by gradient descent. Our method formulate the scene construction process as an attribute searching and image content similarity measurement problem. And we train a generative model to approximate the engine output. As a result, the simulated images fits the distribution of real-world images, and can be used for a downstream task to boost the test performance on real scenes. As far as we know, it is the first attempt to use a differentiable process to finish the dataset synthesis task. By doing this, we hope the work can bring more inspirations for this problem towards using more differentiable methods.

CONFLICT OF INTEREST

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Zhenfeng Xue https://orcid.org/0000-0002-9593-9429
Weijie Mao https://orcid.org/0000-0001-5791-1823

REFERENCES

1. Deng, J., Dong, W., Socher, R., Li, L. & Li, F.F.J.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2009)
2. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision 88(2), 303–338 (2010)
3. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. Int. J. Rob. Res. 32(11), 1231–1237 (2013)
4. Zheng, L., Shen, L., Tian, L., Wang, S., Tian, Q.: Scalable person re-identification: A benchmark. In Int. Conf. Comput. Vis. (2015)
5. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2016)
6. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European Conference on Computer Vision. Springer, Berlin (2016)
7. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2016)
8. Ruiz, N., Schulter, S., Chandraker, M.: Learning to simulate. Paper presented at Seventh International Conference on Learning Representations, New Orleans, 6–9 May 2019
9. Kar, A., Prakash, A., Liu, M.Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., Fidler, S.: Meta-sim: Learning to generate synthetic datasets. In: International Conference on Computer Vision. IEEE, Piscataway (2019)
10. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.: Simulating content consistent vehicle datasets with attribute descent. In: European Conference on Computer Vision. Springer, Berlin (2020)
11. Xue, Z., Mao, W., Zheng, L.: Learning to simulate complex scenes for street scene segmentation. IEEE Trans. Multimedia, to be published, (2021). https://doi.org/10.1109/TMM.2021.3062497
12. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. 13(1), 723–773 (2012)
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. MIT Press, Cambridge (2017)
14. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8(3-4), 229–256 (1992)
15. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2016)
16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. Springer, Berlin (2016)
17. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. In ICML, (2016)
18. Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. Paper presented at Thirty-fifth International Conference on Machine Learning, Stockholm, 10–15 July 2018
19. Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.: Image-to-image translation with conditional adversarial networks (2017)
20. Zhu, J.-Y., Park, T., Isola, P., Efros, A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: International Conference on Computer Vision. IEEE, Piscataway (2017)
21. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.-H.: Universal style transfer via feature transforms. In: Advances in Neural Information Processing Systems, vol. 05, pp. 386–396. MIT Press, Cambridge (2017)
22. Deng, W., Zheng, L., Ye, Q., Kang, G., Yang, Y., Jiao, J.: Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person re-identification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2018)
23. Emami, H., Aliabadi, M.M., Dong, M., Chinnam, R.B.: Spa-gan: Spatial attention gan for image-to-image translation. IEEE Trans. Multimedia (2020)
24. Zhang, Y., Qiu, Z., Yao, T., Liu, D., Mei, T.: Fully convolutional adaptation networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2018)
25. Chen, Y., Li, W., Chen, X., Van Gool, L.: Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach (2019)
26. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680. MIT Press, Cambridge (2014)
27. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. Paper presented at 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, 13–15 November 2017
28. Wrenninge, M., Unger, J.: Synscapes: A photorealistic synthetic dataset for street scene parsing. arXiv preprint arXiv:1810.08705 (2018)
29. Wu, Y., Wu, Y., Gkioxari, G., Tian, Y.: Building generalizable agents with a realistic and rich 3d environment. arXiv preprint, arXiv:1801.02209 (2018)
30. Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., Torralba, A.: Virtualhome: Simulating household activities via programs. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 8494–8502. IEEE, Piscataway (2018)
31. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. Paper presented at 5th International Conference on Learning Representations, Palais des Congrès Neptune, Toulon, France, 24–25 April 2017
32. Devaranjan, J., Kar, A., Fidler, S.: Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In: European Conference on Computer Vision, vol. 11, pp. 715–733. Springer, Berlin (2020)
33. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3907–3916. IEEE, Piscataway (2018)
34. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: IEEE International Conference on Computer Vision (ICCV). IEEE, Piscataway (2019)
35. Genova, K., Cole, F., Maschinot, A., Sarna, A., Vlasic, D., Freeman, W.T.: Unsupervised training for 3d morphable model regression. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8377–8386. IEEE, Piscataway (2018)
36. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. ACM Trans. Graphics 37(6), 1–11 (2018)
37. Loubet, G., Holzschuch, N., Jakob, W.: Reparameterizing discontinuous integrands for differentiable rendering. ACM Trans. Graphics 38(6), 1–14 (2019)
38. Zhang, C., Wu, L., Zheng, C., Gkioulekas, I., Zhao, S.: A differential theory of radiative transfer. ACM Trans. Graphics 38(6), 1–16 (2019)
39. Zhang, C., Miller, B., Yan, K., Gkioulekas, I., Zhao, S.: Path-space differentiable rendering. ACM Trans. Graph. 39(4), 143:1–143:19 (2020)
40. Shi, T., Yuan, Y., Fan, C., Zou, Z., Liu, Y.: Face-to-parameter translation for game character auto-creation. In: International Conference on Computer Vision. IEEE, Piscataway (2019)
41. Chen, D., Lu, Y., Jing, L., Yu, N., Gang, H.: Stylebank: An explicit representation for neural image style transfer. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2017)
42. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. Paper presented at 5th International Conference on Learning Representations, Palais des Congrès Neptune, Toulon, France, 24–25 April 2017
43. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: International Conference on Computer Visison. IEEE, Piscataway (2017)
44. Tian, Q.C., Schmidt, M.: Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337 (2016)
45. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway (2016)

46. Gu, S., Chen, C., Liao, J., Yuan, L.: Arbitrary style transfer with deep feature reshuffle. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Piscataway (2018)
47. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying mmd gans. Paper presented at 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
49. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
50. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. Paper presented at 4th International Conference on Learning Representations, Caribe Hilton, San Juan, Puerto Rico, 2–4 May 2016
51. LeCun, Y., Cortes, C., Burges, C.J.C.: The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/

How to cite this article: Xue, Z., Mao, W., Liu, Y.: Image-level dataset synthesis with an end-to-end trainable framework. IET Image Process. 1–12 (2022). https://doi.org/10.1049/ipr2.12486