RELATE: Physically Plausible Multi-Object Scene Synthesis Using Structured Latent Spaces

Sébastien Ehrhardt [1]* **Oliver Groth**[1]* **Aron Monszpart**[3] **Martin Engelcke**[1]
Ingmar Posner[1] **Niloy Mitra**[2] **Andrea Vedaldi**[1]
[1]Department of Engineering Science, University of Oxford
[2]Department of Computer Science, University College London
[3]Niantic
{hyenal,ogroth}@robots.ox.ac.uk

Abstract

We present RELATE, a model that learns to generate physically plausible scenes and videos of multiple interacting objects. Similar to other generative approaches, RELATE is trained end-to-end on raw, unlabeled data. RELATE combines an object-centric GAN formulation with a model that explicitly accounts for correlations between individual objects. This allows the model to generate realistic scenes and videos from a physically-interpretable parameterization. Furthermore, we show that modeling the object correlation is *necessary* to learn to disentangle object positions and identity. We find that RELATE is also amenable to physically realistic scene editing and that it significantly outperforms prior art in object-centric *scene* generation in both synthetic (CLEVR, ShapeStacks) and real-world data (street traffic scenes). In addition, in contrast to *state-of-the-art* methods in object-centric generative modeling, RELATE also extends naturally to dynamic scenes and generates *videos* of high visual fidelity[2].

1 Introduction

We consider the problem of learning to generate plausible images of scenes starting from parameters that are physically interpretable. Furthermore, we wish to learn such a capability from raw images alone, without any manual or external supervision. Image generation is often approached via Generative Adversarial Networks (GAN) [10]. These models learn to map noise vectors, used as a source of randomness, to image samples. While the resulting images are realistic, the random vectors that parameterize them are not interpretable. To address this issue, authors have recently proposed to *structure* the latent space of deep generative models, giving it a partial physical interpretability [26, 27, 33]. For example, HoloGAN [26] samples volumes and cameras to generate 2D images of 3D objects, and BlockGAN [27] creates scenes by composing multiple objects. The resulting GANs are shown to learn concepts such as viewpoint and object disentangling from raw images.

BlockGAN is of particular interest because, via its relatively strong architectural biases, it provides *interpretable* parameters for the scene, incorporating concepts such as position and orientation. However, BlockGAN comes with a significant limitation in that it assumes that objects are mutually *independent*. This approximation is acceptable only when objects interact weakly, but it is badly violated for medium to densely packed scenes, or for scenes such as stacking wooden blocks or cars following a path, where the (object) correlation is strong.

*indicates equal contribution
[2]See more results at http://geometry.cs.ucl.ac.uk/projects/2020/relate/.

Preprint. Under review.

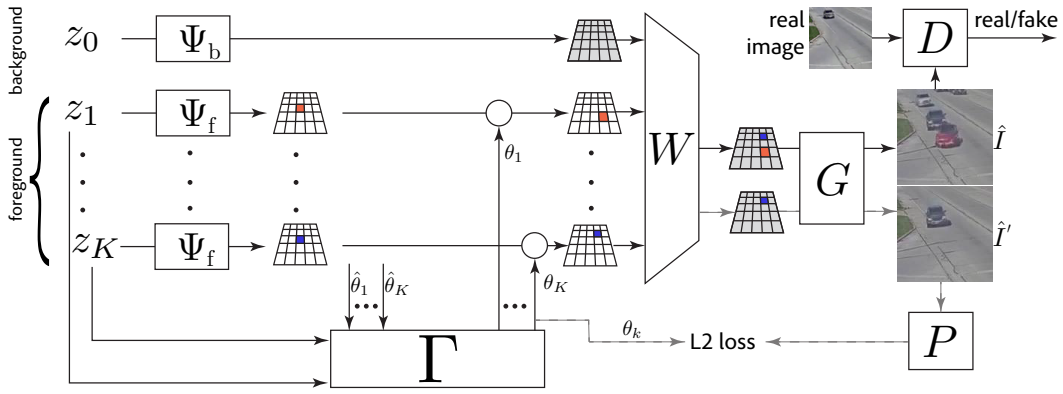


Figure 1: Image generation using RELATE. Individual scene components like background and foreground objects are represented by appearance z_0 and pairs of appearance and pose vectors $(z_i, \theta_i), i \in \{1, \ldots, K\}$, respectively. The key spatial relationship module Γ adjusts the initial independent pose samples $\hat{\theta}_i$ to be physically plausible (*e.g.*, non-intersecting) to produce θ_i. The structured scene tensor W is finally transformed by the the generator network G to produce an image $\hat{I}$. RELATE is trained end-to-end in a GAN setup (D denotes the discriminator and P the position regressor) on real unlabelled images.

Recent work in object-centric generative modeling has attempted to specifically address this by capturing correlations in latent space (*e.g.*, [7, 33]). However as object state information remains significantly entangled in these models they have, to date, been unable to operate on real-world data.

In this paper, we introduce RELATE, a model which explicitly leverages the strong architectural biases of BlockGAN to effectively model correlations between latent object state variables. This leads to a powerful model class, which is able to capture complex physical interactions, while still being able to learn from raw visual inputs alone. Empirically, we show that only when we model such interactions our GAN model correctly disentangles different objects when they exhibit even a moderate amount of correlation (figures 2 and 3). Without this component, the model may still generate high fidelity images, but it generally fails to establish a physically-plausible association between the parameters and the generated images. Our results also demonstrate that GANs are surprisingly sensitive to the correlation of objects in natural scenes, and can thus be used to directly learn these *without* resorting to techniques such as variational auto-encoding (VAE [19]).

We demonstrate the efficacy of RELATE in several scenarios, including balls rolling in bowls of variable shape [6], cluttered tabletops (CLEVR [16]), block stacking (ShapeStacks [12]), and videos of car traffic at a busy intersection. By ablating the interaction module, we show that modeling the spatial correlation between the objects is key. Furthermore, we compare RELATE to several recent GAN- and VAE-based baselines, including BlockGAN [27], GENESIS [7] and OCF [1], in terms of *Fréchet Inception Distance (FID)* [13], and outperform even the best state-of-the-art model by up to 29 points.

Qualitatively, we show that modeling spatial relationships strongly affects scene decomposition and the enforcement of spatial constraints in the generated images. We also show that the physically interpretable latent space learned by RELATE can be used to edit scenes as well as to generate scenes outside the distribution of the training data (*e.g.*, containing more or fewer objects). Finally, we show that the parameterization can be used to generate long plausible video sequences (as measured according to FVD score [31]) by simulating their dynamics while preserving their spatial consistency.

2 Related Work

Interpretable Object-Centric Visual Models. Inspired by the *analysis-by-synthesis* approach for visual perception discussed in cognitive science [41], recent work [3, 7, 11, 33] propose structured latent space models to explain and synthesize images as sets of constituent components which are individually represented using VAEs [19] or GANs [10]. Other approaches favor explicit symbolic representations over distributed ones when parsing an image [30, 37] or propose probabilistic programming languages to formalize image generation [22]. In both cases, object-centric modeling

allows decomposition of images into components and also enables targeted image modification via interpolation in symbol or latent space, *e.g.*, altering position or color of an object parsed by the model. Interpretable and controllable factors of image generation are desirable properties for neural rendering models and have been investigated in recent image generation models, *e.g.*, [1, 24, 26, 27]. However, despite the modeling effort put into the object representations, inter-object interactions are typically only modeled in a less explicit way, *e.g.*, via image layers [33, 37], depth ordering variables [1] or an autoregressive *scene prior* [7]. Our work adds to this line of work by proposing a spatial correlation network which facilitates disentanglement of learned object representations and can be trained from raw observations.

Neural Physics Approximation. Harnessing the power of deep learning to approximate physical processes is an emerging trend in the machine learning community. Especially the approximation of rigid body dynamics with neural networks already boasts a large body of literature, *e.g.*, [2, 4, 8, 21, 32, 36]. Such learned approximations of object interactions have been successfully employed in object manipulation [15] and tracking [9, 20]. However, most entries in this line of work are only applied to visual toy domains or rely on segmentation masks or bounding boxes to initialize their object representations before the networks approximate the object dynamics. While we leverage ideas from neural dynamics modeling, we go beyond the established scope of visual toy domains such as colored point masses or moving MNIST digits [23] and learn directly from rich visual data such as simulated object stacks and real traffic videos without further annotation.

Extraction and Generation of Video Dynamics. Videos are a natural choice of data source to learn about the physics of rigid bodies. Physical information extracted from videos can either be explicit such as estimates of velocity or friction [39] or implicitly represented in the latent space, *e.g.*, sub-spaces corresponding to pose variation [5]. More recently, object-centric approaches have also been leveraged to acquire better video representations for future frame prediction [40] or model-based reinforcement learning [34]. Several studies have also attempted to learn entire video distributions as spatio-temporal tensors in GAN frameworks [17, 29, 35, 38] yielding impressive first results for full video generation in artificial and real domains. In contrast to prior art, our model departs from a monolithic spatio-temporal tensor representation over an entire video. Instead we cast the video learning and generation process as temporal extension of the object-centric representation of a single frame, lowering the computational burden while still faithfully representing long-range dynamics.

3 Method

RELATE (figure 1) consists of two main components: An interaction module, which computes physically plausible inter-object and object-background relationships, and a scene composition and rendering module, which features an interpretable parameter space factored into appearance and position vectors. The details are given next.

3.1 Physically-interpretable scene composition and rendering

RELATE considers scenes containing up to K distinct objects. The model starts by sampling *appearance parameters* $z_1, \dots, z_K \sim \mathcal{U}([-1, 1]^{N_f})$ for each individual foreground object as well as a parameter $z_0 \sim \mathcal{U}([-1, 1]^{N_b})$ for the background. These parameters are small noise vectors, similar to the ones typically used in generative networks. Different from the object poses below, they are sampled independently, thus assuming that the appearance of different objects is independent.

For rendering an image, the appearance parameter z_k is first mapped to a tensor $\Psi_k \in \mathbb{R}^{H \times H \times C}$. This is done via two separate learned decoder networks, one for the background $\Psi_0 = \Psi_b(z_0)$ and one for the foreground objects $\Psi_k = \Psi_f(z_k)$. Here H is the horizontal and vertical spatial resolution of the representation (see table A1) and C is the number of feature channels (see tables A2 and A3). Since we assume that individual objects are much smaller than the overall scene, we restrict Ψ_k, $k \geq 1$ to be non-zero only in a fixed window of size $H' < H$ in the center of the tensor.

Each foreground object also has a corresponding *pose parameter* θ_k, which is geometrically interpretable. For simplicity, we assume $\theta_k \in \mathbb{R}^2$ to be a 2D translation, acting on the tensor Ψ_k via bilinear resampling:

$$\hat{\Psi}_k = \theta_k \cdot \Psi_k \quad \text{such that} \quad [\hat{\Psi}_k]_u = [\Psi_k]_{u+\theta_k}$$

where $u \in \mathbb{R}^2$ is a spatial index and $[\cdot]_u$ means accessing the column of the tensor in bracket at spatial location u (using padding and bilinear interpolation if u does not have integer coordinates). However, θ_k can easily be extended to represent full 3D transformations as previously shown in BlockGAN [27].

Foreground and background objects are composed into an overall scene tensor $W \in \mathbb{R}^{H \times H \times C}$ via element-wise max- (or sum-) pooling as $W_u = \max_{k=0,\dots,K} [\hat{\Psi}_k]_u$. In this manner, the scene tensor is a function $W(\Theta, Z)$ of the pose parameters $\Theta := (\theta_1, \dots, \theta_K)$ and the appearance parameters $Z := (z_0, z_1, \dots, z_K)$. Finally, a decoder network $\hat{I} = G(W)$ renders the composed scene as an image (see table A4).

Discussion. This model is 'physically interpretable' in the sense that it captures (1) the identities of K distinct objects and (2) their pose parameters as translation vectors. This should be contrasted to traditional GAN models, where the code space is given as an uninterpretable, monolithic noise vector z. Despite the structure given to the code space, there is no guarantee that the model will actually learn to map it to the corresponding structure in the example images. However, we found empirically that this is the case as long as the correlations between the different objects are also captured.

3.2 Modeling correlations in scene composition

RELATE departs significantly from prior art such as BlockGAN as it does not assume the parameters θ_i of the different objects to be independent. In order to model correlation, we propose a two-step procedure, based on a residual sampler. First, we sample a vector of K i.i.d. poses $\hat{\Theta} \sim \mathcal{U}([-H''/2, H''/2]^{2K})$ where $H'' < H$ is smaller than the spatial size H of the tensor encoding. Then, we pass this vector to a 'correction' network Γ that remaps the initial configuration to one that accounts for the correlation between object locations and appearances, as well as between objects and the background (coded by the appearance component z_0 in z): $\Theta := \Gamma(\hat{\Theta}, Z)$. In practice, we expect object interactions, as any physical law, to be *symmetric* with respect to the order of the objects. We obtain this effect by implementing Γ as running K copies of the *same* corrective function in parallel:

$$\theta_k = \hat{\theta}_k + \zeta(\hat{\theta}_k, z_k, |z_0, \{z_i, \hat{\theta}_i\}_{i \geq 1, i \neq k}). \tag{1}$$

The function ζ is implemented in a manner similar to the Neural Physics Engine (NPE) [4]:

$$\zeta(\hat{\theta}_k, z_k, |z_0, \{z_i, \hat{\theta}_i\}_{i \geq 1, i \neq k}) = f(\hat{\theta}_k, z_k, z_0, h_k^s), \quad h_k^s = \sum_{q \neq k} g(\hat{\theta}_k, z_k, \hat{\theta}_q, z_q), \tag{2}$$

where f and g are Multi Layer Perceptrons (MLPs) (tables A5, A6) operating on stacked vector inputs and h^s is an embedding capturing the interactions between the K objects. Besides symmetry, an advantage of this scheme is that it can take an arbitrary number of objects K due to the sum-pooling operator used to capture the interactions. In this manner, the sampler Γ is automatically defined for any value of K, which is sampled uniformly from an interval $[K_{\text{min}}, K_{\text{max}}]$. Furthermore, sampling independent quantities followed by a correction has the benefit of injecting some variance on the objects' positions at the early stage of training, which helps to avoid converging to trivial/bad solutions.

Ordered scenes. An advantage of RELATE is that it can be easily modified to take advantage of additional structure in the scene. For scenes where objects have natural order, such as stacks of blocks, we experiment with conditioning pose θ_i on the preceding pose θ_{i-1}, using a Markovian process. This is done by first sampling $\hat{\theta}_1 \sim \mathcal{U}([-H''/2, H''/2])$, and then applying a correction to account for the background z_0 as before, finally sampling the other objects in sequence:

$$\theta_1 = \hat{\theta}_1 + f_0(\hat{\theta}_1, z_1, z_0), \qquad \forall k > 1: \quad \theta_k = \theta_{k-1} + f_1(\theta_{k-1}, z_{k-1}, z_0), \tag{3}$$

where f_0, f_1 are implemented as MLPs as before (tables A9, A10). Note that this can be interpreted as a special case of the model above in the sense that we can write $\Theta := \Gamma(\hat{\Theta}, Z)$, provided that $\hat{\theta}_k = 0$ for $k \geq 2$.

Modeling dynamics. RELATE can also be immediately extended to make dynamic predictions. For this, we sample the initial positions $\theta_k(0)$ as before and then update them incrementally as $\theta_k(t+1) = \theta_k(t) + v_k(t+1)$, where $v_k(t)$ is the object velocity. In order to obtain the latter, we let $V_k(t) = [v_k(t-i)]_{i=2,1,0}$ denote the last three velocities of the k-th object. The initial

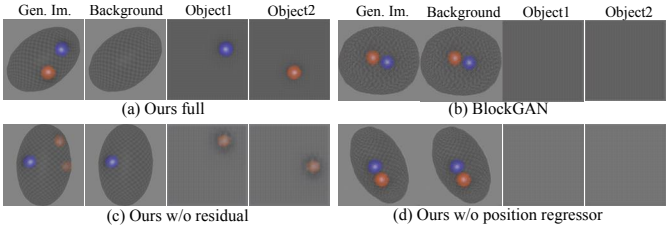


Figure 2: Ablation Study. For every case we render every component of our method independently. We show that only our full model is able to correctly disentangle individual components of the scene.

Table 1: Ablation study. FID score (lower is better) on BALLSINBOWL. Ours (full) reaches the highest fidelity by a large margin.

BlockGAN 2D[27]	149.8
Ours w/o residual	137.1
Ours w/o pos. reg.	147.8
Ours (full)	**77.4**

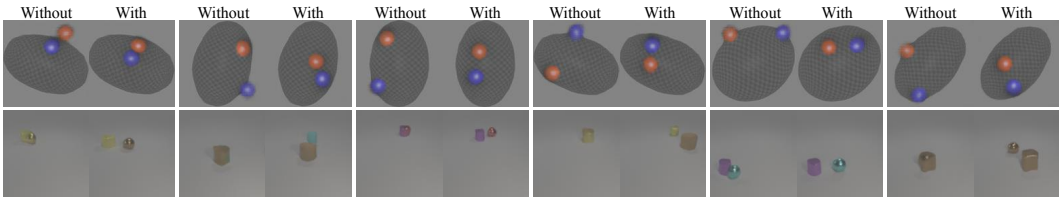


Figure 3: Effect of the interaction module Γ. We show pairs of images without and with the correction function Γ is applied. For BALLSINBOWL the correction moves the balls within the bowl, and for CLEVR it pushes apart intersecting objects.

value $V_k(0) = e_v(z_k, z_0, \theta_k(0))$ is initialized as a function of the appearance parameters and initial positions (table A7); and we use the NPE style update equations [4], where e_v, f_v and g_v are MLPs,

$$v_k(t+1) = f_v(\theta_k(t), z_k, V_k(t), z_0, h_k^d(t)), \quad h_k^d(t) = \sum_{q \neq k} g_v(\theta_k(t), z_k, V_k(t), \theta_q(t), z_q, V_q(t)). \quad (4)$$

3.3 Learning objective

Training our model makes use of a training set I_i, $i = 1, \ldots, N$ of N images of scenes containing different object configurations. No other supervision is required. Our learning objective is a sum of *two high fidelity losses* and *a structural loss* which we describe below.

For high fidelity, images $\hat{I}$ generated by the model above are contrasted to real images I from the training set using the standard GAN discriminator $\mathcal{L}_{\text{GAN}}(\hat{I}, I)$ and style $\mathcal{L}_{\text{style}}(\hat{I}, I)$ losses from [27].

In addition, we introduce a regularizer to encourage the model to learn a non-trivial relationship between object positions and generated images. For this, we train a position regressor network P that, given a generated image $\hat{I}$, predicts the location of the objects in it. In practice, we simplify this task and generate an image $\hat{I}'$ by retaining only object k of the K objects at random and minimizing $\|\check{\theta}_k - P(G(W(z_0, z_k, \theta_k)))\|_2^2$. Here the symbol $\check{}$ means that gradients are not back-propagated through θ_k: this is to avoid mode collapse of the position at zero. P shares most of its weights with the discriminator network (see table A8).

In the case of dynamic prediction, the discriminator takes as input the sequence of images concatenated along the RGB dimension and is tasked to discriminate between fake and real sequences. Similar to a static model we also have a position regressor which is tasked to predict the position of an object rendered at random with zero velocity.

4 Experiments

Implementation details. We learn mappings Ψ_b and Ψ_f using the same Adaptive Instance Normalization (AdaIN) [14] architecture. The spatial size of their output tensors is set to $H = 16$ and the final output image to 128×128 (which is reduced when needed for fair comparison to other methods). We used the Adam [18] optimizer for learning and train for a fixed number of epochs and always select the last model snapshot. We consider GENESIS [7] and OCF [1] baselines, quoting results from the original papers whenever possible, and BlockGAN2D as an ablation of our method.

Table 2: Comparison to state-of-the-art methods. FID score (lower is better) for various datasets. We constistently outperfom prior art in object centric scene generation. 'Ordered' refers to the variant discussed in sec. 3.2. Note that on ShapeStacks, BlockGAN2D collapses to render all towers in the background and is not object-centric therefore (see fig. A3).

RELATE variant	CLEVR-5 General	CLEVR-5vbg General	CLEVR General	ShapeStacks Ordered	REALTRAFFIC General
OCF [1]	N/A	83.1	N/A	N/A	N/A
GENESIS [7]	211.7	169.4	151.3	233.0	167.1
BlockGAN2D * [27]	73.2	63.8	94.4	105.1	57.9
Ours	**70.5**	**54.3**	**79.7**	**102.3**	**42.0**

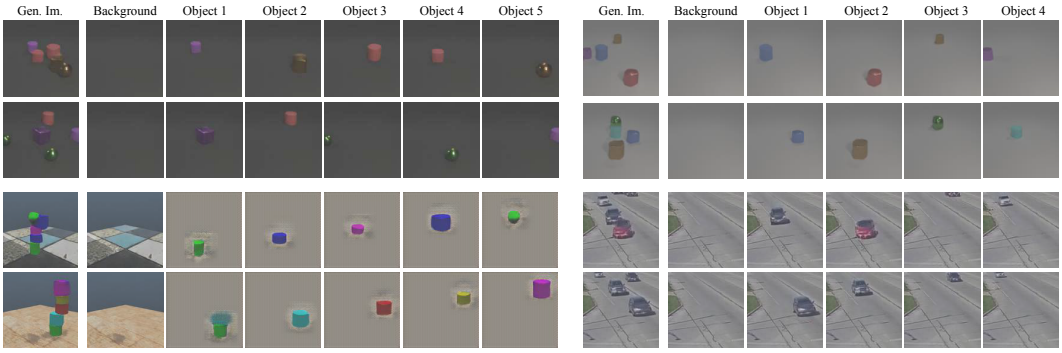


Figure 4: Component-wise scene generation. From a generated image (left) RELATE can render each component individually for each dataset. For CLEVR and REALTRAFFIC objects are rendered after being composed with the background (cf. section 4.2). Top left picture has increased contrast for easier visualization.

Datasets. We conduct experiments on four different datasets. First, we consider a relatively simple dataset, BALLSINBOWL from [6], for assessing the model features and ablations. This data consists of videos of two distinctly colored balls rolling in an elliptical bowl of variable orientation and eccentricity. Interactions amount to object collisions and the fact that they must roll within the bowl. To this, we add two popular synthetic datasets CLEVR [16] (cluttered tabletops) and ShapeStacks [12] (block stacking). Finally, we collected a new dataset REALTRAFFIC containing five hours of footage of a busy street intersection, divided into fragments containing from one to six cars. Especially the last dataset contains many interactions between the individual cars as they adapt their speed to the surrounding traffic which happens frequently when the light changes and cars either slow down because of a queue on red or accelerate when the lights change to green again. Further details about training, evaluation and datasets can be found in the appendix sections A2, A3 and A5.

4.1 Generating static scenes

Ablation study. We start experimenting with the comparatively simple BALLSINBOWL dataset to conduct basic ablations. The first ablation removes the spatial correlation module Γ and the position regression loss, therefore reducing RELATE to a 2D version of BlockGAN. We also consider *'w/o residual'*, where the addition $\hat{\theta}_k$ in equation (4) is removed, and *'w/o pos. loss'*, where the position regression loss regularizer is removed. Table 1 shows that each component of RELATE yields an improvement in terms of FID scores on this dataset supporting our spatial modeling decisions. Furthermore, in figure 2 we show qualitatively that only RELATE (a) is able to correctly disentangle the underlying scene factors. We do this by generating the same image while retaining a single factor, which correctly isolates the background, and, in turn, both individual objects. BlockGAN 2D (b) and *'ours w/o pos. reg.'* (d) fail to disentangle the factors entirely, mapping everything to the background component. *'Ours w/o residual'* (c) shows that the model partially fails to disentangle, with the background encoding some but not all the objects. Finally, figure 3 visualizes the effect of the interaction module Γ. Recall that this is implemented as a 'correction' function that accounts for correlation starting from independently-sampled parameters. For BALLSINBOWL, the correction module moves the balls within the bowl, and for the CLEVR it pushes objects apart if they intersect.

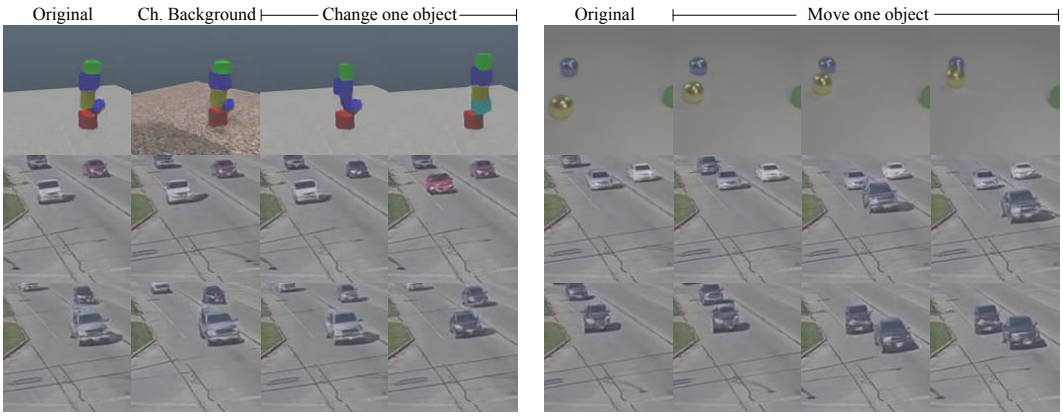


Figure 5: Image editing. Left: We demonstrate the capacity of RELATE to change the background and the appearance of individual objects. Right: RELATE is also able to modify the position of a single object.

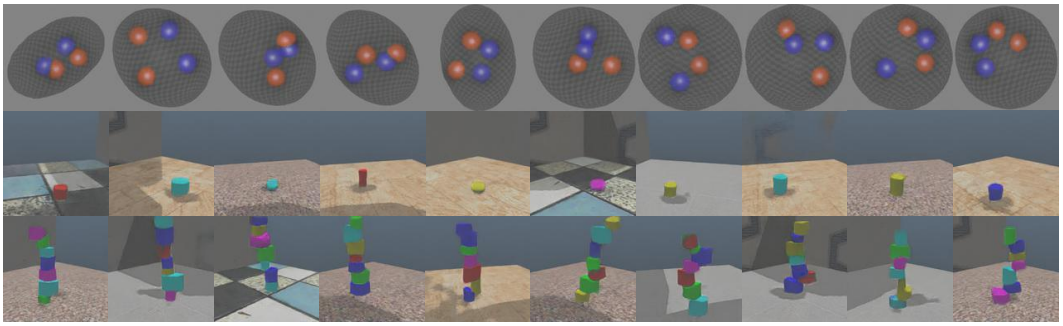


Figure 6: Out-of distribution generation. RELATE can generate images outside the training distribution. The first row shows generating a bowl with four balls, whereas the training set only features exactly two. The last two rows depict towers of one and seven objects, whereas the training images only had stacks of height two to five.

Quantitative evaluation. In table 2, we compare RELATE to existing scene generators on ShapeStacks, CLEVR and REALTRAFFIC. We report performance in terms of FID [13] score between 10,000 images sampled from our model and the respective test sets. For CLEVR, we train RELATE and BlockGAN on a restricted version of the data containing from three to six objects in an image[3], and at test time, we require all models to sample images with three to ten objects. We consistently outperform all prior methods in all scenes and scenarios according to FID scores. In particular, on CLEVR, RELATE can generate a larger number of objects than seen during training suggesting its improved generalization capabilities which are demonstrated further in figure 6.

4.2 Interpretability of the latent space and scene editing

As shown in the ablation studies in figure 2, RELATE successfully disentangles a scene into independent components - in contrast to BlockGAN2D which struggles to separate individual objects from the background. Figure 4 shows that RELATE can *disentangle* also far more complex scenes in REALTRAFFIC, ShapeStacks and CLEVR. Note that for REALTRAFFIC and CLEVR we render objects composed with the background. In fact in these datasets the size and appearance of each object is correlated to their position in the background because of the camera perspective. Next, in figure 5 we use RELATE to *edit* a generated scene. For example we can change the position or identity of individual objects. Finally, we show that RELATE can generate *out-of-distribution* scenes. This is achieved in particular by sampling a different number of objects. In figure 6, for instance, RELATE is trained on ShapeStacks seeing towers of height two to five. However, it can render taller towers of up to seven objects, or even just a single object. Likewise, in BALLSINBOWL it can generate bowls with four balls having seen only two during training. Furthermore, in ShapeStacks

[3]Note that GENESIS was trained on the full training set featuring three to ten objects.

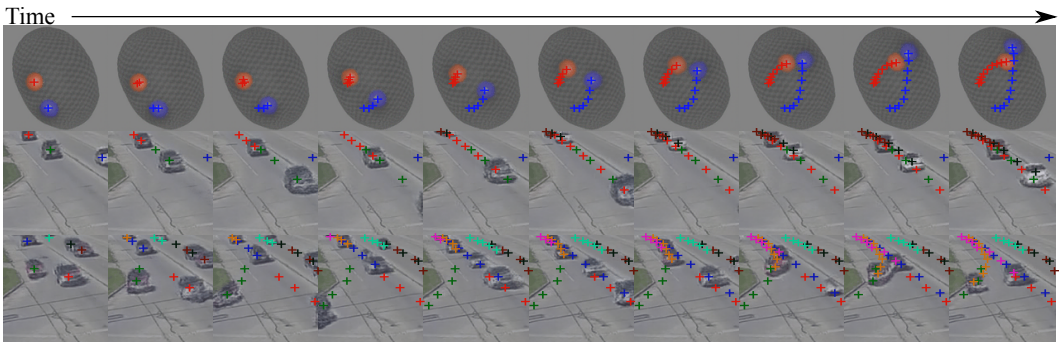


Figure 7: Video generation. We show consecutive video frames generated by RELATE overlayed with crosses representing projections of the model's estimated pose parameters for each object. In BALLSINBOWL the interaction with the environment is well captured as the balls stay within the bowl. In REALTRAFFIC the cars stay in their lane, or can decide to make a right turn (last row).

Figure 8: Failure case. Our model struggles to understand big changes of aspect ratio. For instance, in the static case, when we drag two cars towards the bottom of the image we can see that the left grey car disappears at some point (highlighted in red) and reappears on the edge of the image. This explains why the video generation on REALTRAFFIC fails to capture the true data distribution more faithfully resulting in lower FVD score.

each tower is composed of blocks of *different* colors, but RELATE can relax this constraint rendering objects with repeated colors.

4.3 Simulating dynamics

We train this model on BALLSINBOWL and REALTRAFFIC to predict 15 and 10 consecutive frames respectively. During generation, we sample videos with a sequence length of 30 frames and measure the faithfulness with respect to the distribution of the test data via the *Fréchet Video Distance (FVD)* [31]. We achieve FVD scores of 556 and 2253 respectively. This is perceptibly better than 920 and 3370 for a baseline consisting of time-shuffled sequences from the respective training sets, which feature perfect resolution but poor dynamics. Qualitatively in figure 7 we see that the model does understand the motion and captures interaction with the background. For instance in BALLSINBOWL the balls do have a curved motion because of the shape of the bowl and decrease in speed when reaching the edges of the bowl which are in higher position (see first row of figure 7). In REALTRAFFIC the cars do stay in their respective lane. Interestingly our model is able to handle different types of motions correctly (see third row figure 7) and use the sample vector to decide whether the cars should go straight or turn. Finally we see that we can also generate videos with much more cars than the upper bound (5) with which the system was trained (see last row figure 7).

Limitations While the dynamics in video generation look realistic in most cases, REALTRAFFIC also exposed the limitations of our approach. In this dataset the perspective range of the camera is important. As a result, cars at the bottom of the image, which appear bigger in the training data, often do not get generated properly in the static case (see highlighted frames in figure 8). We hypothesize that this is also the main reason why the cars' appearance (and hence FVD score) deteriorates in the dynamic scenarios: since the style parameters z_i are fixed to preserve identity, it is not possible for the model to account accurately for the appearance change introduced by large changes of perspective over the course of a sequence. To verify our hypothesis in appendix A4 we proposed a simple modification of our pipeline that accounts for scale changes in an image. This modification simply allows the network to predict H' for each individual object instead of it being a constant. We found that in general it allows our network to train on datasets with more important range of scales such as CLEVR3 [1] and renders objects at more different scales for CLEVR and REALTRAFFIC (see figure A1). Besides in table A11 we show that this modification almost always results in higher

FID scores from our main model as well as a significant boost of 397 points in FVD score for REALTRAFFIC (1855 vs 2253).

5 Conclusion

We have introduced RELATE, a GAN-based model for object-centric scene synthesis featuring a physically interpretable parameter space and explicit modeling of spatial correlations. Our experimental results suggest that spatial correlation modeling plays a pivotal role in disentangling the constituent components of a visual scene. Once trained, RELATE's interpretable latent space can be leveraged for targeted scene editing such as altering object positions and appearances, replacing the background or even inserting novel objects. We demonstrate our model's effectiveness by presenting *state-of-the-art* scene generation results across a variety of simulated and real datasets. Lastly, we show how our model naturally extends to the generation of dynamic scenes being able to generate entire videos from scratch. A main limitation of our current model is its restriction to planar motions which prevents it from representing arbitrarily 3D motions featuring angular rotation more faithfully, most notably highlighted by the experiments for video generation. However, we believe that our work can contribute to future research in that area by providing a scalable spatio-temporal modeling approach which is conveniently trainable on unlabeled data.

6 Acknowledgments

This work is supported by the European Research Council under grants ERC 638009-IDIU and ERC 677195-IDIU. The authors acknowledge the use of Hartree Centre resources in this work. TheSTFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. The authors also acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work (`http://dx.doi.org/10.5281/zenodo.22558`). The authors also thank Olivia Wiles for feedback on the paper draft.

Broader Impact

Our method advances the ability of computers to learn to understand environments in images in an object-centric way. It also enhances the capabilities of generative models to generate realistic images of "invented" environment configurations.

Overall, we believe our research to be at low to no risk of direct misuse. At present, our generation results are insufficient to fool a human observer. However, it has to be noted that the sampling process is, as in many other deep generative models, capable of revealing patterns observed in the training data, *e.g.*, specific textures or object geometries. Such data privacy concerns are not applicable in the street traffic data used in our research, since the resolution of the videos is far too low to identify individual drivers or recognize cars' license plates. However, 'training data leakage' should be taken into consideration when the model is trained on more sensitive datasets.

In a positive prospect, we believe that our model contributes to further the development of less opaque machine learning models. The explicit object-centric modelling of image components and their geometric relationships is in many of its aspects intelligble to a human user. This facilitates debugging and interpreting the model's behaviour and can help to establish trust towards the model when employed in larger application pipelines.

However, the key value of our paper is in the methodological advances. It is conceivable that, like any advance in machine learning, our contributions could ultimately lead to methods that in turn can and are misused. However, there is nothing to indicate that our contributions facilitate misuse in any direct way; in particular, they seem extremely unlikely to be misused directly.

References

[1] Titas Anciukevicius, Christoph H Lampert, and Paul Henderson. Object-centric image generation with factored depths, locations, and appearances. *arXiv preprint arXiv:2004.00642*, 2020.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[3] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[4] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2017.

[5] Emily L Denton and vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4414–4423. Curran Associates, Inc., 2017.

[6] Sebastien Ehrhardt, Aron Monszpart, Niloy J Mitra, and Andrea Vedaldi. Taking visual motion prediction to new heightfields. *Computer Vision and Image Understanding*, 181:14–25, 2019.

[7] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020.

[8] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. In *International Conference on Learning Representations*, 2016.

[9] Fabian B Fuchs, Adam R Kosiorek, Li Sun, Oiwi Parker Jones, and Ingmar Posner. End-to-end recurrent multi-object tracking and trajectory prediction with relational reasoning. *arXiv preprint arXiv:1907.12887*, 2019.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. 2014.

[11] Klaus Greff, Raphaël Lopez Kaufmann, Rishab Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[12] Oliver Groth, Fabian B Fuchs, Ingmar Posner, and Andrea Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 702–717, 2018.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

[14] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[15] Michael Janner, Sergey Levine, William T. Freeman, Joshua B. Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-oriented prediction and planning. In *International Conference on Learning Representations*, 2019.

[16] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[17] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1771–1779. JMLR.org, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[20] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616, 2018.

[21] Jannik Kossen, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting. Structured object-aware physics prediction for video modeling and planning. In *International Conference on Learning Representations*, 2020.

[22] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015.

[23] Yann LeCun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[24] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. *arXiv preprint arXiv:1912.05237*, 2019.

[25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. 2018.

[26] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7588–7597, 2019.

[27] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. BlockGAN: Learning 3d object-aware scene representations from unlabelled images. *arXiv preprint arXiv:2002.08988*, 2020.

[28] Danilo J Rezende and Fabio Viola. Generalized elbo with constrained optimization, geco. In *Workshop on Bayesian Deep Learning, NeurIPS*, 2018.

[29] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[30] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[31] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[32] Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations*, 2018.

[33] Sjoerd van Steenkiste, Karol Kurach, Jürgen Schmidhuber, and Sylvain Gelly. Investigating object compositionality in generative adversarial networks. *CoRR*, abs/1810.10340, 2019.

[34] Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *International Conference on Learning Representations*, 2019.

[35] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[36] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in neural information processing systems*, pages 4539–4547, 2017.

[37] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 699–707, 2017.

[38] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 91–99. Curran Associates, Inc., 2016.

[39] Tian Ye, Xiaolong Wang, James Davidson, and Abhinav Gupta. Interpretable intuitive physics model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[40] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10353–10362, 2019.

[41] Ilker Yildirim, Max Siegel, and Joshua Tenenbaum. 34 physical object representations. *The Cognitive Neurosciences*, pages 399–409, 2020.

Supplementary Material for "RELATE: Physically plausible Multi-Object Scene Synthesis Using Structured Latent Spaces"

In this supplementary material we provide further details about RELATE. We organized this manuscript as follow. We first describe the losses we use in more details in appendix A1. We follow with a more in-depth description of implementation details in appendix A2. After, appendix A3 is dedicated to explain in more details how the various baselines were trained. Appendix A4 explains in more details the scale augmented model reference in section 4. Appendix A5 contains a more thorough explanation of every dataset and data collection when applicable. Finally we provide more qualitative results in appendix A6.

A1 Losses

Our final loss is the sum of three losses mentioned in the main text:

$$\mathcal{L}_{tot} = \mathcal{L}_{\text{GAN}}(\hat{I}, I) + \mathcal{L}_{\text{style}}(\hat{I}, I) + \min_{G,\Gamma,P} \|\breve{\theta}_k - P(G(W(z_0, z_k, \theta_k)))\|_2^2,$$

where $\mathcal{L}_{\text{GAN}}(\hat{I}, I)$ is the standard GAN loss:

$$\mathcal{L}_{\text{GAN}}(\hat{I}, I) = \min_{G,\Gamma} \max_{D} \mathbb{E}[\log(1 - D(G(W(Z, \Theta)))] + \mathbb{E}[\log(D(I))]$$

The style loss follows the implementation of BlockGAN [27]. The input of the style discriminator D_l are mean μ_l and variance σ_l^2 across spatial dimensions of $\Phi_l(x) \in \mathcal{R}^{W_l \times H_l \times C_l}$, the output of the lth layer of D taken before the normalization step:

$$\mu_l(\Phi_l(x)) = \frac{1}{W_l \times H_l} \sum_i \sum_j \Phi_l(x)_{i,j},$$

$$\sigma_l^2(\Phi_l(x)) = \frac{1}{W_l \times H_l} \sum_i \sum_j (\Phi_l(x)_{i,j} - \mu_l(\Phi_l(x)))^2.$$

The style discriminator D_l for each layer is then implemented as a linear layer followed by a sigmoid activation function. The resulting style loss is:

$$\mathcal{L}_{\text{style}}(\hat{I}, I) = \max_{D} \sum_l \mathbb{E}[\log(1 - D_l(\hat{I}))] + \mathbb{E}[\log(D_l(I))]$$

A2 Implementation details

Infrastructure and framework For all experiments we use PyTorch 1.4. We train all models on a single NVIDIA Tesla V100 GPU.

Training hyperparameters We initialize all weights (including instance normalization ones) by drawing from a random normal distribution $\mathcal{N}(0, 0.02)$. All biases are initialized to 0. For each update of the discriminator we update the generator M number of times. We use Adam parameters $(\beta_1, \beta_2) = (0., 0.999)$ for all datasets except BALLSINBOWL where $\beta_1 = 0.5$. Similarly W was a max-pooling operator in all datasets except BALLSINBOWL where we used a sum pooling operator. As in BlockGAN[27], background and foreground decoders each start from a learned constant tensors T_b and T_f respectively with sizes $H \times H \times 256$ and $H' \times H' \times 512$. For BALLSINBOWL we use a tensor T_f for each object and use a constant style vector of one.

In the case of dynamic scenarios we reuse same hyperparameters as in the static case except that we use a learning rate of 0.0001 and $\beta_1 = 0$.

Full details of the parameters for each dataset can be found in table A1

Evaluation details For FID scores computation we draw 10 000 samples from our model which we compare against the same number of images drawn from the test set. To compute FVD score on each dataset, we sample 500 videos of 30 frames from our model and compare them against the videos of the respective test sets (500 for BALLSINBOWL and 275 videos for REALTRAFFIC). This also applies to the time shuffled baseline.

Table A1: Hyper-parameters for each datasets. Epoch nums are the number of epochs we trained for.

Dataset	Learning rate	Epoch nums	M	$K_{min} - K_{max}$	H'	N_b	N_f	H''/H sampling range
BALLSINBOWL	0.001	60	1	2-2	8	3	1	$[-0.8, 0.8]^2$
CLEVR5	0.0001	40	2	2-5	4	1	90	$[-0.6, 0.6]^2$
CLEVR5-vbg	0.0001	30	2	2-5	4	1	90	$[-0.6, 0.6]^2$
CLEVR	0.0001	40	2	3-6	4	1	90	$[-0.6, 0.6]^2$
ShapeStacks	0.001	30	2	2-5	4	12	64	$[-0.6, 0.6] \times [0, 0.6]$
REALTRAFFIC	0.0001	20	2	1-5	6	1	20	$[-0.6, 0.6]^2$

Table A2: Network architecture for the foreground object generator Ψ_f.

Layer name	Layer Type	Input size	Output size	Kernel Size	Stride	Activation	Norm.
Style_f	Id	$H' \times H' \times 512$	$H' \times H' \times 512$	-	-	Id	AdaIn
Convtf_1	ConvTranspose	$H' \times H' \times 512$	$H' \times H' \times 512$	3×3	1	LeakyReLU	AdaIn
Convtf_2	ConvTranspose	$H' \times H' \times 512$	$H' \times H' \times 256$	3×3	1	LeakyReLU	AdaIn
Pad	Padding	$H' \times H' \times 512$	$H \times H \times 256$	-	-	-	-

To be able to compare with other methods we resize our generated images to 96×96 on CLEVR5 and CLEVR5-vbg and 64×64 for ShapeStacks. We evaluate on the generated 128×128 images otherwise.

We empirically found that background was rendered with better quality for lower values of z_0. Hence at test time we sampled z_0 from $\mathcal{U}([-0.5, 0.5]^{N_b})$ for optimal background fidelity rendering.

A2.1 Architecture details

Generator. In this work we maintain the core of our architecture fixed as much as possible. Since the dimension of the sample z_i does not necessarily match the channel dimension where it is injected before applying Adaptive Instance Normalisation (AdaIN) to a layer l we map z_i to a vector $\hat{z}_i$ transformed such that

$$\hat{z}_i = \max(W_l^T z_i + b_l, 0)$$

Where (W_l, b_l) are learnable parameters. AdaIN is applied at the end of the layers (after the activation). All LeakyReLU layers are using a parameter of 0.2.

Discriminator We describe the architecture of the discriminator network in more details in table A8. We use spectral normalization [25] at almost every layer. Positions are directly regressed from the last feature output of the discriminator (see last line P_{end}). Therefore in practice P and D share the same backbone D_b (see table table A8 until flatten) for every image I:

$$P(I) = P_{end}(D_b(I)), \qquad D(I) = Disc(D_b(I)).$$

Input for style discriminator are taken after the convolution of (Convd_2, Convd_3, Convd_4, Convd_5) in table A8 before the normalization. Spectral Normalization was *not* applied to any D_l.

Table A9: Network architecture for module f_0.

Layer name	Layer Type	Input size	Output size	Activation
FCf_0_1	Linear	$N_f + N_b + 2$	128	LeakyReLU
FCf_0_2	Linear	128	64	LeakyReLU
FCf_0_3	Linear	64	2	Tanh

Table A10: Network architecture for module f_1.

Layer name	Layer Type	Input size	Output size	Activation
FCf_1_1	Linear	$N_f + N_b$	128	LeakyReLU
FCf_1_2	Linear	128	64	LeakyReLU
FCf_1_3	Linear	64	2	None
Pos_{out}	Sigmoid(x) Tanh(y)	2	2	None

Table A3: Network architecture for the background object generator Ψ_b.

Layer name	Layer Type	Input size	Output size	Kernel Size	Stride	Activation	Norm.
Style_b	Id	$H \times H \times 256$	$H \times H \times 512$	-	-	Id	AdaIn
Convtb_1	ConvTranspose	$H \times H \times 512$	$H \times H \times 512$	3×3	1	LeakyReLU	AdaIn
Convtb_2	ConvTranspose	$H \times H \times 512$	$H \times H \times 256$	3×3	1	LeakyReLU	AdaIn

Table A4: Network architecture for the generator G. Outputs of all K foreground object generators Ψ_f and background generator Ψ_b are stacked on the first dimension before entering layer W (third row).

Layer name	Layer Type	Input size	Output size	Kernel Size	Stride	Activation
Ψ_f (see table A2)	-	$H' \times H' \times 512$	$16 \times 16 \times 256$	-	-	-
Ψ_b (see table A3)	-	$H \times H \times 512$	$16 \times 16 \times 256$	-	-	-
W	Max/Sum Pool	$(K+1) \times 16 \times 16 \times 256$	$16 \times 16 \times 256$	-	-	-
Convtg_1	ConvTranspose	$16 \times 16 \times 256$	$32 \times 32 \times 128$	4×4	2	LeakyReLU
Convtg_2	ConvTranspose	$32 \times 32 \times 128$	$64 \times 64 \times 64$	4×4	2	LeakyReLU
Convtg_3	ConvTranspose	$64 \times 64 \times 64$	$64 \times 64 \times 64$	3×3	1	LeakyReLU
Convtg_4	ConvTranspose	$64 \times 64 \times 64$	$128 \times 128 \times 64$	4×4	2	LeakyReLU
Convtg_5	ConvTranspose	$128 \times 128 \times 64$	$128 \times 128 \times 3$	3×3	1	Tanh

A3 Baselines

OCF. OCF results were copied from original paper of [1].

BlockGAN2D. We use the same hyperparameters and network architecture as RELATE except for learning rate and M. In all cases we report the best results over models trained with variations of learning rate in (0.001, 0.0001) and M in (2,3).

GENESIS. We use the official implementation[4] of GENESIS for all experiments. For the ShapeStacks dataset, we use the official model snapshot released with the original paper[5]. For all other datasets, we train GENESIS for 500,000 iterations with the default learning parameters and select the last model checkpoint for evaluation. When training GENESIS we use *constrained ELBO optimization* [28] controlled via `g_goal` in the training script which influences the decomposition capability of GENESIS. We perform a grid search over `g_goal` in the range of 0.5635 to 0.5655 and select the model with the lowest ELBO after 500,000 iterations.

A4 Scale augmented model

To tackle the limitation discussed in section 4.3 we propose to augment our model with scale prediction. Practically this sums up to predict H' for each individual objects. Therefore we now assign H'_k instead of H' to each foreground component of the scene. H'_k is computed thanks to a module sc following the equation: $H'_k = H' \times (1 + sc(z_0, \theta_k, z_k))$, sc is further described in table A12. We summarize all hyper-parameters used for the training of the model in table A13. For evaluation we sample z_0 from $\mathcal{U}([-1, 1]^{N_b})$, expect for REALTRAFFIC video model where we used $\mathcal{U}([-0.5, 0.5]^{N_b})$ where this range was optimal for background fidelity.

A5 Datasets

BALLSINBOWL. This dataset is a replica of the two balls synthetic dataset of [6]. It consists of 2500 training sequences and 500 test sequences of two balls of different fixed colour rolling in bowls of various shapes. We count an epoch as 10,000 iterations over the data. In figure A2 we show some sample data from this dataset.

[4] `https://github.com/applied-ai-lab/genesis`

[5] `https://drive.google.com/drive/folders/1uLSV5eV6Iv4BYIyh0R9DUGJT2W6QPDkb?usp=sharing`

Table A5: Network architecture for module f and f_v. * indicates modification of f_v

Layer name	Layer Type	Input size	Output size	Activation
FCf_1	Linear	$2 \times (N_f + 2 + 2^*)$	32	LeakyReLU
FCf_2	Linear	32	32	LeakyReLU
FCf_3	Linear	32	32	None

Table A6: Network architecture for module g and g_v. * indicates modification of g_v

Layer name	Layer Type	Input size	Output size	Activation
FCg_1	Linear	$32 + N_f + 2 + 2^* + N_b$	32	LeakyReLU
FCg_2	Linear	32	32	LeakyReLU
FCg_3	Linear	32	2	Tanh

CLEVR. We used the official CLEVR from [16]. We train on data from train and validation set and evaluate on the test set. Both ours and BlockGAN2D were trained on the subset containing 3 to 6 objects and evaluated on the entire test set.

CLEVR3/CLEVR5/CLEVR5-vbg. We use online code provided by the authors[6] to generate CLEVR3, CLEVR5 and CLEVR5-vbg. As done in [1] we generate 100,000 images keep 90,000 for training and 10,000 for testing.

ShapeStacks We use the official release of the ShapeStacks dataset[7]. We use the `default` partitioning provided with the dataset and merge the training and validation splits for a total of 264,384 training images. All FID comparisons are made against 10,000 images randomly sampled from the test set which contains 46,560 images in total. Since the original resolution of the images is 224×224 pixels, we re-scale them to 128×128 before feeding them to our network.

REALTRAFFIC. We recorded 5 hours from Youtube[8] of a live traffic camera at a crossing. The video was then unrolled at 10 fps and manually processed to keep only sequences with a number of cars in [1,5]. We kept 560 videos for the training set and 123 in test (80/20 ratio). This dataset will be publicly released.

A6 Qualitative results

We provide additional qualitative generation results. Figure A3 shows a failure case of BlockGAN2D mentioned in the paper. In fact, when the scene is more structured BlockGAN2D fails to be object centric and let the background render the entire scene. In addition figures A4, A6 and A7 to A9 provide more samples on every dataset for all the models we trained. In particular we can see that when inter-objects relations are weak in CLEVR5 or CLEVR5-vbg, BlockGAN2D performs qualitatively similar to ours (see figures A6 and A7). However when the scene is more crowded and the objects have higher correlation BlockGAN2D quality decreseases significantly (see figures A4, A8 and A9).

[6] `https://github.com/TitasAnciukevicius/clevr-dataset-gen.`

[7] `https://shapestacks.robots.ox.ac.uk/#data`

[8] `https://www.youtube.com/watch?v=5_XSYlAfJZM`

Table A7: Network architecture for module e_v.

Layer name	Layer Type	Input size	Output size	Activation
FCe_v_1	Linear	$N_f + 2 + N_b$	128	LeakyReLU
FCe_v_2	Linear	128	128	LeakyReLU
FCe_v_3	Linear	128	3×2	Tanh

Table A8: Network architecture for the discriminators. Note that the Instance Normalization weights were also subjected to spectral normalization. P and D shares weights until Flatten layer.

Layer name	Layer Type	Input size	Output size	Kernel Size	Stride	Activation	Norm.
Convd_1	Conv	$128 \times 128 \times 3$	$64 \times 64 \times 64$	5×5	2	LeakyReLU	-
Convd_2	Conv	$64 \times 64 \times 64$	$32 \times 32 \times 128$	5×5	2	LeakyReLU	IN/Spec. Norm.
Convd_3	Conv	$32 \times 32 \times 128$	$16 \times 16 \times 256$	5×5	2	LeakyReLU	IN/Spec. Norm.
Convd_4	Conv	$16 \times 16 \times 256$	$8 \times 8 \times 512$	5×5	2	LeakyReLU	IN/Spec. Norm.
Convd_5	Conv	$8 \times 8 \times 512$	$4 \times 4 \times 1024$	5×5	2	LeakyReLU	IN/Spec. Norm.
Flatten	Id	$4 \times 4 \times 1024$	$1 \times 1 \times 16384$	-	-	-	-
$Disc$	Linear	$1 \times 1 \times 16384$	1	-	-	Sigmoid	None/Spec. Norm.
P_{end}	Linear	$1 \times 1 \times 16384$	2	-	-	Tanh	None/Spec. Norm.

Table A11: Comparison of RELATE to its scale augmented version (see appendix A4). FID score (lower is better) for various datasets. X indicates failure of convergence. Overall predicting scale improves FID score except in the REALTRAFFIC dataset where there is a marginal decrease of performance. For CLEVR-3, which features wider of objects sizes, our augmented model is the only one to reach convergence.

RELATE variant	CLEVR-3 General	CLEVR-5 General	CLEVR-5vbg General	CLEVR General	ShapeStacks Ordered	REALTRAFFIC General
Ours	X	70.5	**54.3**	79.7	102.3	**42.0**
Ours + scale	**49.3**	**56.7**	**54.3**	**58.3**	**91.8**	46.8

Table A12: Network architecture for module sc.

Layer name	Layer Type	Input size	Output size	Activation
FCsc_1	Linear	$N_f + 2 + N_b$	32	LeakyReLU
FCsc_2	Linear	32	32	LeakyReLU
FCsc_3	Linear	32	1	Tanh

Table A13: Hyper-parameters for each datasets for the scale augmented model. Epoch nums are the number of epochs we trained for. Model is described in appendix A4.

Dataset	Learning rate	Epoch nums	M	$K_{min} - K_{max}$	H'	N_b	N_f	H''/H sampling range
CLEVR3	0.0001	40	2	2-3	6	1	20	$[-0.6, 0.6]^2$
CLEVR5	0.0001	40	2	2-5	4	1	20	$[-0.6, 0.6]^2$
CLEVR5-vbg	0.0001	40	2	2-5	4	1	20	$[-0.6, 0.6]^2$
CLEVR	0.0001	40	2	3-6	6	1	20	$[-0.6, 0.6]^2$
ShapeStacks	0.001	30	2	2-5	4	5	20	$[-0.6, 0.6] \times [0, 0.6]$
REALTRAFFIC	0.0001	20	2	1-5	6	1	20	$[-0.6, 0.6]^2$

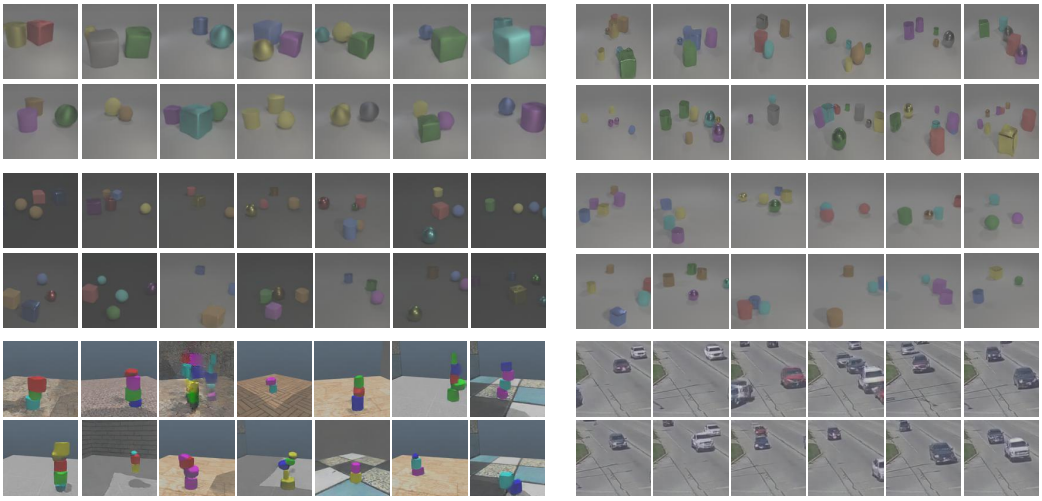


Figure A1: Samples from the scale augmented RELATE. From left to right and top to bottom we display CLEVR-3, CLEVR, CLEVR-5vbg, CLEVR-5, ShapeStacks and REALTRAFFIC. The model makes high fidelity prediction in the case of CLEVR-3 while the objects' scale varies a lot between different images. Similarly predictions on CLEVR comprise more variety of object sizes compared to our main model (see figure A8). Finally on REALTRAFFIC we can see that cars can be rendered at arbitrary points in the road.

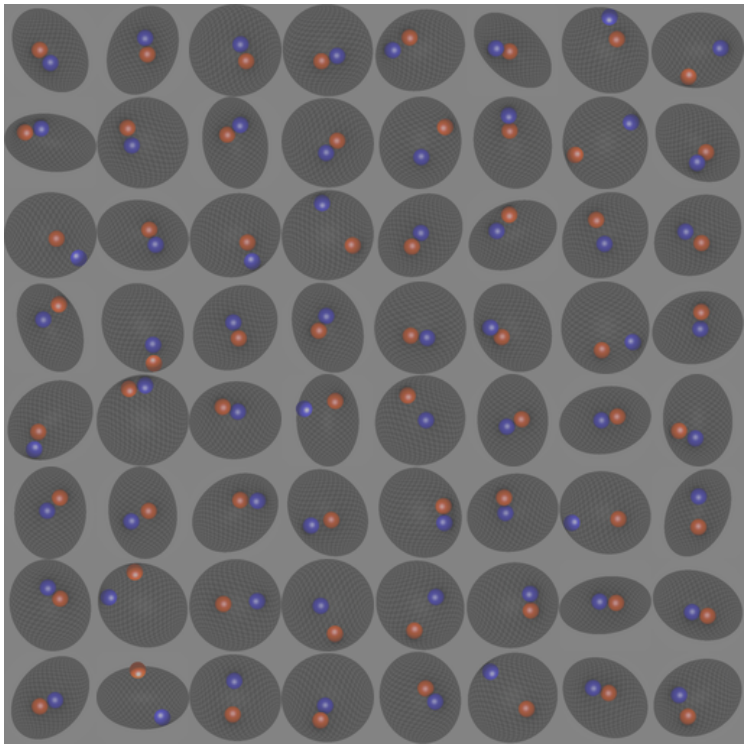


Figure A2: Sample data from BALLSINBOWL. The dataset consists of two balls of different colors rolling in elliptical bowls of various shapes.

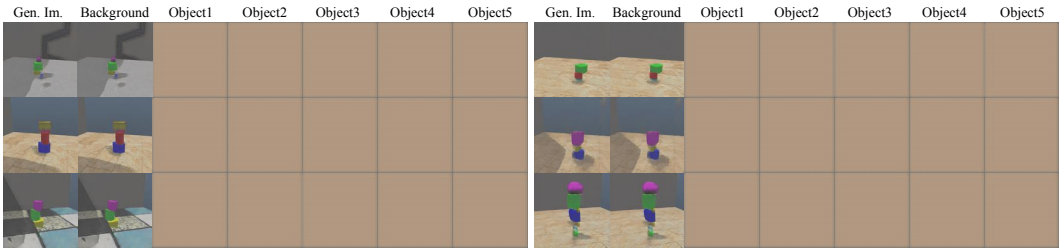
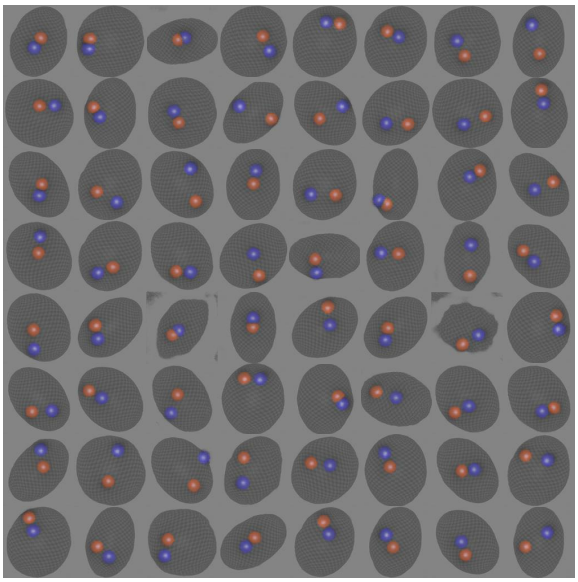
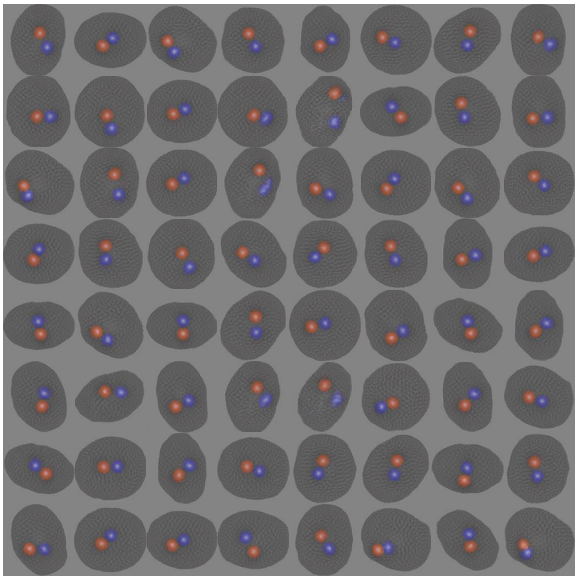


Figure A3: BlockGAN2D scene decomposition on ShapeStacks. We display in order (generated, background, object_1, ..., object_5) for BlockGAN2D model. We see that in this case BlockGAN does not decompose the scene into objects and renders everything in the background instead. This shows how, for structured scenes, prior work fails to capture correlations between objects.

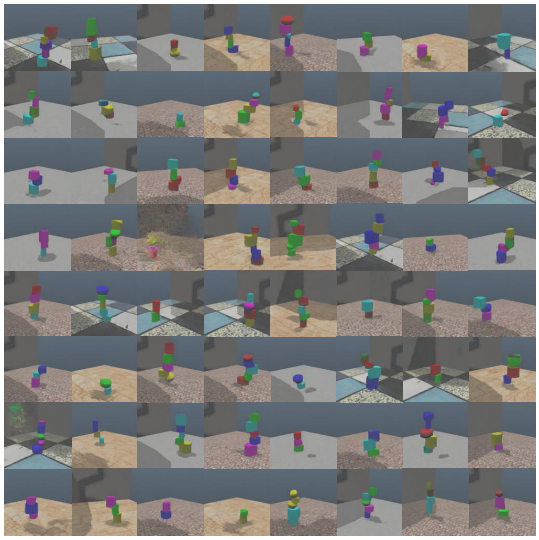


RELATE

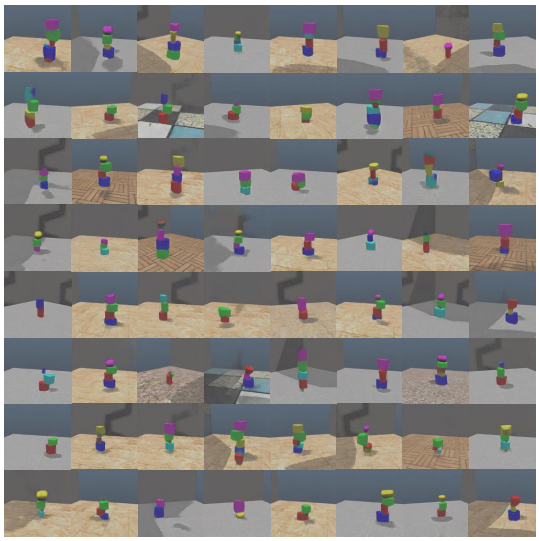


BLOCKGAN2D

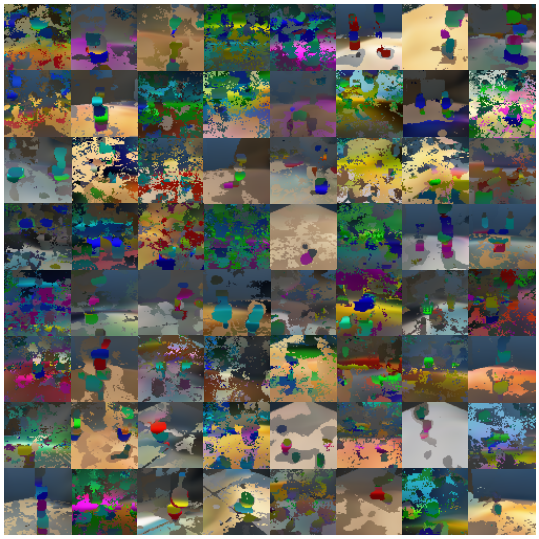
Figure A4: Generated scenes for models trained on BALLSINBOWL. Qualitatively RELATE generates images of higher quality compared to BlockGAN2D[27].



RELATE

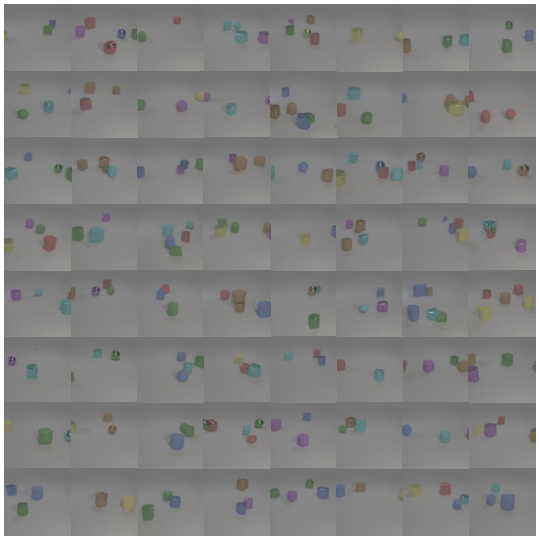


BLOCKGAN2D

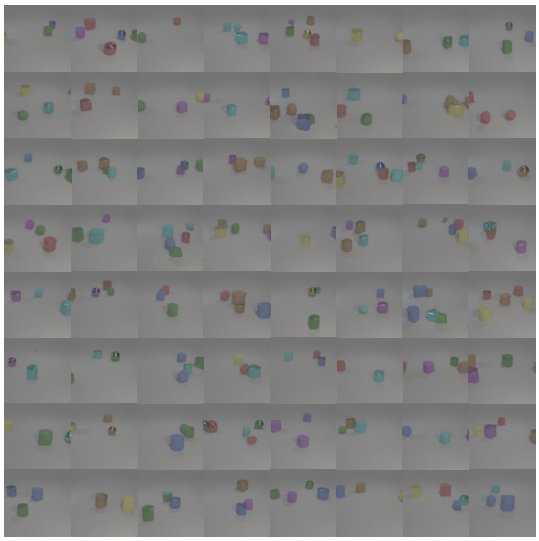


GENESIS

Figure A5: Generated scenes for models trained on ShapeStacks. Despite qualitative similar rendering, BlockGAN2D does not render a scene component-wise as opposed to ours (see figure A3).



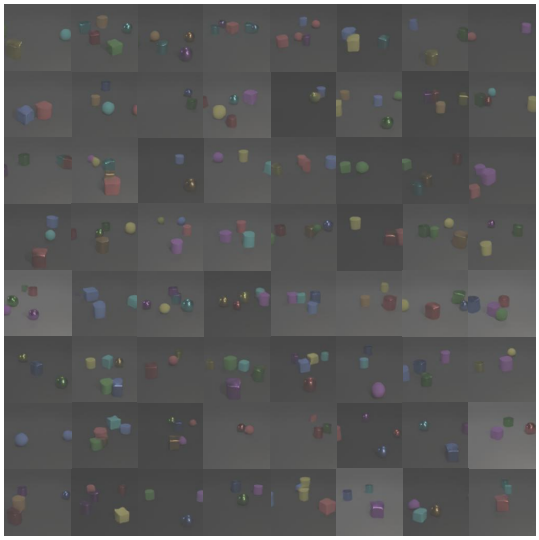
RELATE



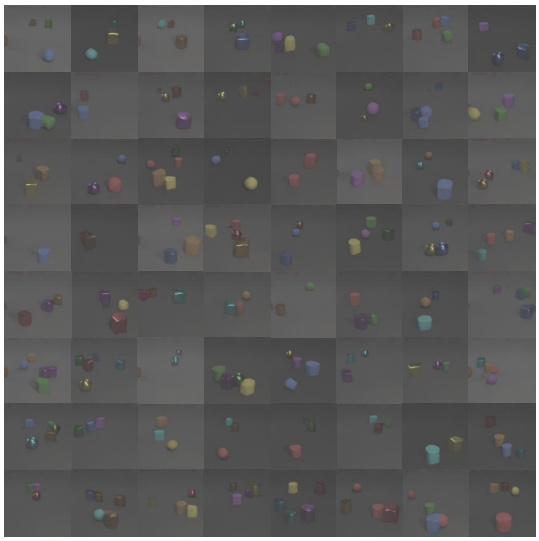
BLOCKGAN2D

GENESIS

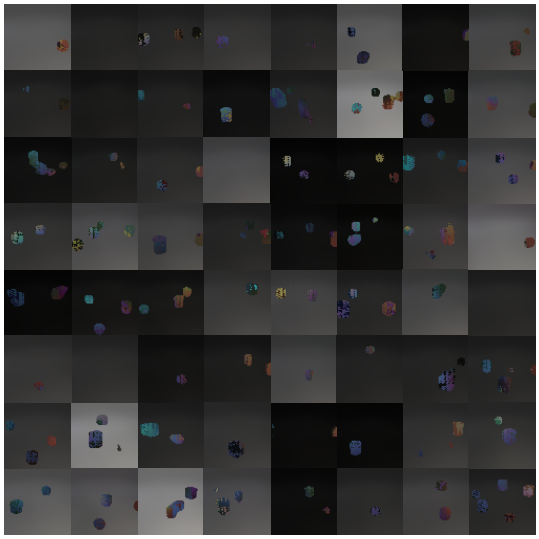
Figure A6: Generated scenes for models trained on CLEVR5. For less crowded scenes our model and BlockGAN2D reach similar performances.



RELATE



BLOCKGAN2D



GENESIS

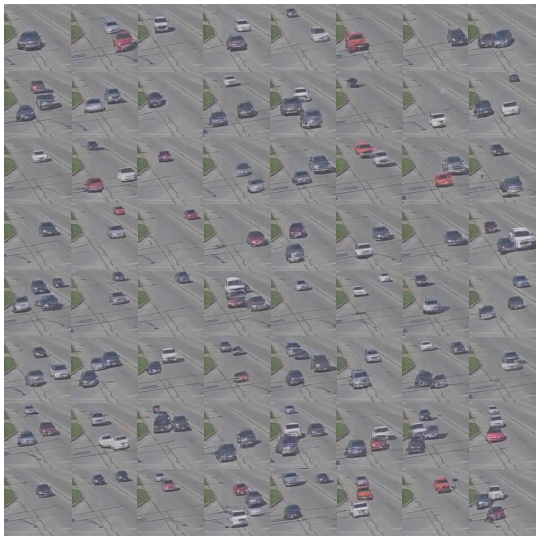
Figure A7: Generated scenes for models trained on CLEVR5-vbg. This scenario reaches similar conclusion as figure A6.

RELATE

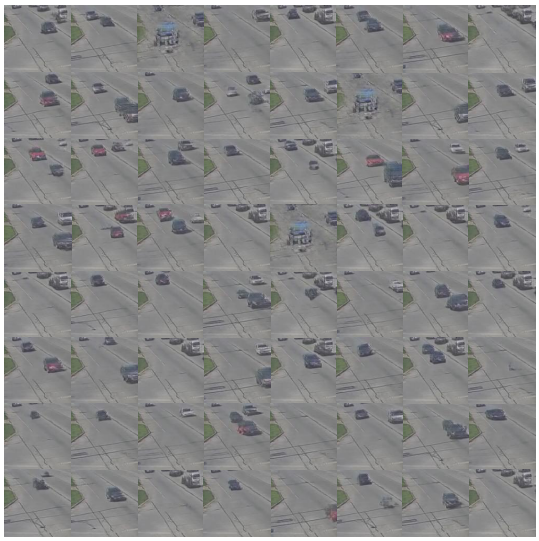
BLOCKGAN2D

GENESIS

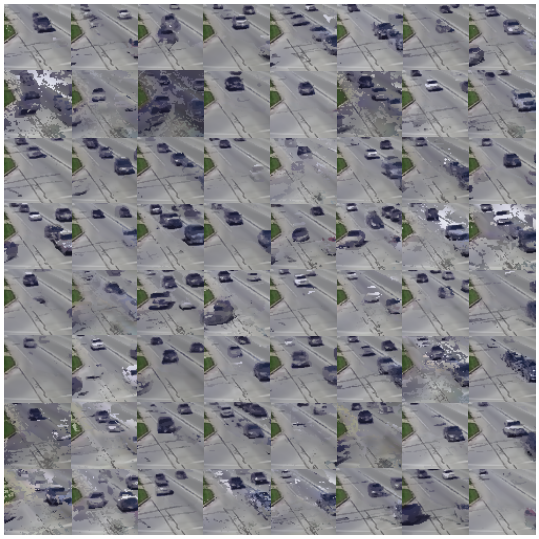
Figure A8: Generated scenes for models trained on CLEVR. When the scene gets more crowded RELATE gets an advantage as it can push objects apart resulting in higher qualitative rendering.



RELATE



BLOCKGAN2D



GENESIS

Figure A9: Generated scenes for models trained on REALTRAFFIC. Our model qualitatively renders higher fidelity images. BlockGAN2D sometimes suffers from background mode collapse (see first, second and fourth rows of second block).