OBJECT-ORIENTED REPRESENTATION OF 3D SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose a generative model, called ROOTS (Representation of Object-Oriented Three-dimension Scenes), for unsupervised object-wise 3D-scene decomposition and and rendering. For 3D scene modeling, ROOTS bases on the Generative Query Networks (GQN) framework, but unlike GQN, provides object-oriented representation decomposition. The inferred object-representation of ROOTS is 3D in the sense that it is viewpoint invariant as the full scene representation of GQN is so. ROOTS also provides hierarchical object-oriented representation: at 3D global-scene level and at 2D local-image level. We achieve this without performance degradation. In experiments on datasets of 3D rooms with multiple objects, we demonstrate the above properties by focusing on its abilities for disentanglement, compositionality, and generalization in comparison to GQN.

1 INTRODUCTION

The shortcomings of contemporary deep learning such as interpretability, sample efficiency, ability for reasoning and causal inference, and transferability and compositionality, are where the symbolic AI has traditionally shown its strengths. One of the grand challenges in machine learning is thus to make deep learning embrace the benefits of symbolic representation so that symbolic entities can emerge from a high-dimensional observation like a visual scene.

Particularly, for learning from visual observations of the physical world, such representation should consider the following criteria. First, it should focus on *objects* (and their *relations*). Objects and relations are the foundational entities of the physical world and can be considered as the basic unit for which we can build a factorized model. Representation supporting such object-wise factorization will help compositionality and transferability by making the object representation a reusable module. Second, being *three-dimensional* (3D) is a decisive property of physical world and thus the representation should also reflect it. We humans, equipped with such 3D representation in our brain, can retain consistency on the identity of an observed object even if it is observed from different viewpoints which result in quite different appearances of the object. Besides, we can infer and imagine the whole scene from these limited partial perspectives. Lastly, learning such representation should be *unsupervised*. Although there have been remarkable advances in supervised approaches to object perception (Redmon et al., 2016; Ren et al., 2015; Long et al., 2015), it should be learned without supervision as we humans do. This unsupervised approach not only avoids expensive labeling efforts, but also allows adaptability and flexibility to downstream task's goals. This is important because the definition of "objectness" itself can vary depending on the situation. In addition, unsupervised learning should help better generalization for new types of objects.

In this paper, we propose a probabilistic generative model that can learn, without supervision, object-oriented representation from partial 2D observations of a 3D scene. We call the proposed model ROOTS (Representing Object-Oriented Three-dimensional Scenes). To this end, we base our model on the Generative Query Networks (GQN) (Eslami et al., 2018). The GQN, as a conditional latent variable model, provides the methodological framework to learn a latent representation of a 3D scene from 2D observations and the corresponding viewpoints. However, unlike GQN which provides a representation encoding a whole 3D scene into a single continuous vector, the scene representation of ROOTS is factorized into object representations per object exiting in the 3D scene while modeling a background representation separately for the remaining part. Thus, from 2D images containing multiple objects with occlusion and partial observation, ROOTS can extract and build a model of an object that is interpretable, composible, and transferable. In addition, ROOTS also provides the object-factorized representation *hierarchically*: one for a global 3D scene and another for local 2D

images. In experiments, we show the above abilities of ROOTS for the 3D-Room dataset which contains images of 3D rooms with several objects with different colors and shapes.

The contributions of the paper are: (i) We proposed a generative model that detects objects, extract background, and learn representation of object-wise 3D model from 2D partial observations. (ii) This is the first unsupervised model that can identify objects in a 3D scene. (iii) We demonstrate various advantages of the extracted and learned object representation including interpretability, transferability, and compatibility.

2 PRELIMINARY: GENERATIVE QUERY NETWORKS

Generative Query Networks (GQN) is a conditional latent variable model mapping an query viewpoint, $x \in \mathbb{R}^{d_x}$, to its corresponding 2D projections, $y \in \mathbb{R}^{d_y}$, under 3D settings. For example, given a set of *context* observations $C = (X_C, Y_C) = \{(x_c, y_c)\}_{c \in \mathcal{I}(C)}$ and *Target* observations $D = (X, Y) = \{(x_i, y_i)\}_{i \in \mathcal{I}(D)}$, GQN is asked to model a conditional prior on the latent variable $p(z|C, x)$. Here, $\mathcal{I}(\mathcal{S})$ stands for the set of data-point indices in a dataset $\mathcal{S}$. For brevity, we however assume in the following that D contains only one pair (x, y) unless mentioned otherwise. Generalization to the multi-instance case is straightforward. Then, the generative process can be written as follows: $p(y|x, C) = \int p(y|z)p(z|C, x)\mathrm{d}z$. In the dataset $\{(C, D)\}$, a pair of context C and target D are assumed to be sampled from the same 3D scene while different pairs are sampled from different 3D scenes. The design principle underlying this modeling is to infer the target scene from the context in such a way that sampling z from $P(z|C, x)$ corresponds to a representation explaining the scene. Due to the intractable posterior, the model is trained via variational approximation which gives the following evidence lower bound (ELBO) objective:

$$\log p_\theta(y|x, C) \geq \mathbb{E}_{q_\phi(z|C,D)}\left[\log p_\theta(y|x, z)\right] - \mathbb{KL}(q_\phi(z|C, D) \parallel p_\theta(z|C)). \tag{1}$$

This ELBO is optimized using the reparameterization trick (Kingma & Welling, 2013).

In the original GQN, the prior is conditioned on the query viewpoint in addition to the context, i.e., $p(z|x, r_C)$, and thus results in inconsistent samples across different viewpoints when modeling uncertainty in the scene. The Consistent GQN (Kumar et al., 2018) (CGQN) resolved this by removing the dependency on the query viewpoint from the prior. This encourage z to be a summary of a full 3D scene independent of the query viewpoint. Hence, it is consistent across viewpoints and more similar to the original Neural Processes. For the remainder of the proposal, we use the abbreviation *GQN* for CGQN unless stated otherwise.

Due to the complexity of 3D scenes, inferring the latent representation of a 3D scene requires more model complexity. For this, GQN uses ConvDRAW (Gregor et al., 2016), an auto-regressive density estimator performing $p(z|C) = \prod_{l=1}^{L} p(z^l|z^{<l}, r_C)$ where L is the number of auto-regressive rollouts and r_C is a pooled context representation $\sum_{c \in \mathcal{I}(C)} f_r(x_c, y_c)$ with f_r an context encoding.

3 ROOTS: REPRESENTATION OF OBJECT-ORIENTED 3D SCENES

As we seek learning 3D representations that can generate a scene observation for any given viewpoint (although the representation is object-oriented), we base our model on the GQN framework. The main advancements of our proposed model are two-fold. First, instead of an encoding compressing the whole scene, we obtain a representation of a scene factorized into objects and background. Second, the object-wise representation is hierarchically structured at two-levels: from global-3D representation to local-2D representation, and then from the local-2D representation to image pixels. The generative process can be explained from these perspectives.

3.1 GENERATIVE PROCESS

Object-Wise Factorization is a process obtaining object-factorized representation $z = \{z_k\}_{k=1}^{K}$ for K objects present in a target 3D space. Because the number of objects K is a random variable, the conditional prior can be written as follows: $p(z|C) = p(K|C)\prod_{k=1}^{K} p(z_k|C)$. To implement such model with variable number of objects, AIR (Eslami et al., 2016) proposed to use an RNN that rolls out K steps, processing one object per step. However, according to our preliminary results and

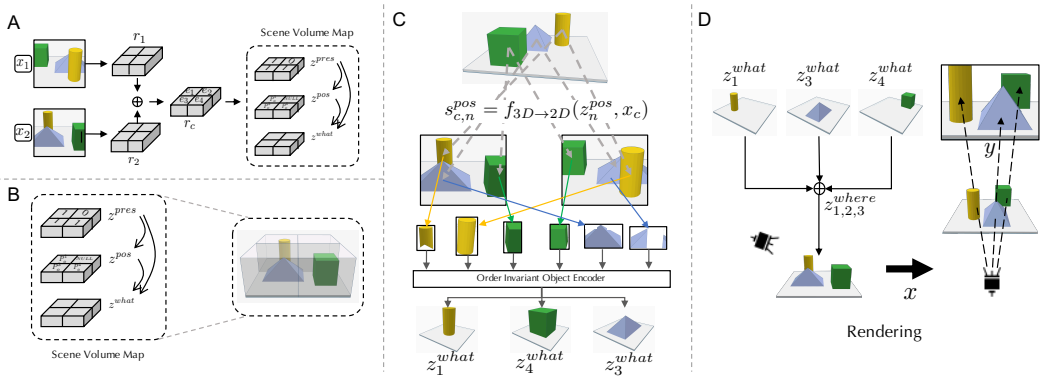

Figure 1: Overview of ROOTS. (A) The *context* observations are first fed into an encoder and obtain the scene-volume map. (B) The scene-volume map has a cell of $z_n = z_n^{pres}, z_n^{pos}, z_n^{what}$ for each spatial volume cell of the target environment. (C) After obtaining z_n^{pres}, z_n^{pos}, we obtain z_n^{what} by finding object patches from context and then clustering them for each object. The resulting representation z_n^{what} is not a 3D-representation. (D) Decoding process $p(x|s)p(s|z,x)$. The representation is rearranged according to query viewpoint x by using $f_{3D\to 2D}(z_n^{pos}, x)$ and then projected to a target 2D image. In (D), left-bottom is an example of 3D full view. A projection camera is shown on the left corner. On the right, we perform 2 steps of projections: (1) converting the coordinate with respect to the projection camera (bottom) and (2) projecting onto 2D canvas (up).

other works (Crawford & Pineau, 2019), it turns out that this is computationally inefficient due to the sequential nature and shows performance degradation as the number of objects increases beyond a few objects.

As a better approach would process objects in a spatially parallel way, we instead introduce the *scene-volume map*. The scene-volume map, denoted by $\mathcal{Z}$, is a tensor of $(H \times W \times L)$-dimension that represents the actual volume of the target 3D space by $N = H \times W \times L$ unit cells. Here, $H, W,$ and L are the three axes (height, width, and layer) of the 3D space, respectively. Our goal is then to associate each volume cell $n \in \mathcal{Z}$ with a latent vector $z_n = (z_n^{pres}, z_n^{pos}, z_n^{what})$ where z_n^{pres} a binary variable indicating the presence of an object in the cell, z_n^{pos} an object's center coordinates in the 3D space, and z_n^{what} a d-dimension vector for representing the appearance of an object. Note that z_n^{pos} and z_n^{what} are defined only when $z_n^{pres} = 1$.

One key idea here is that introducing a presence variable per volume cell is to reflect the inductive bias of physics: two objects cannot co-exist at the same position. This helps remove the sequential object processing as dealing with an object does not need to consider other objects if the features are already from spatially distant areas. With this parallelization, learning is easier and faster. Note also that the presence variable represents the existence of a center point in a volume cell, not the full volume of an object. Thus, the actual volume of an object can still exist across neighboring cubes. This results in the following prior model $p(z|C) = p(K|C)\prod_{k=1}^{K} p(z_k|C) =$

$$\prod_{n=1}^{N} p(z_n|C) = \prod_{n=1}^{N} p(z_n^{pres}|C)p(z_n^{pos}|C, z_n^{pres})p(z_n^{what}|C, z_n^{pres}, z_n^{pos}) \tag{2}$$

where $p(z_n^{pos}|C, z_n^{pres}) = p(z_n^{pos}|C)^{z_n^{pres}}$ and $p(z_n^{what}|C, z_n^{pres}, z_n^{pos}) = p(z_n^{what}|C, z_n^{pos})^{z_n^{pres}}$, and we have $K = \sum_{n=1}^{N} z_n^{pres}$.

Hierarchical Object-Oriented Representation. In ROOTS, the representation is not only factorized object-wise but also hierarchical. That is, at the first level of the hierarchy, we obtain the object-wise 3D representation, which is viewpoint-invariant, for a global-3D scene. This is global as we model here all existing objects in the target 3D space regardless of any specific viewpoint. At the second level, we then introduce the local-2D object-wise representation $s = \{s_n\}$ which is the latent factors of the generated image and thus viewpoint-dependent. Because of this, some global object n may not present in the local-2D representation s, i.e., if the object is out of the view field from a given viewpoint. Then, the full generative process of ROOTS can be written as:

$$p(y, s, z|x, C) = p(y|s)p(s|z, x)p(z|C) = p(y|s)\prod_{n=1}^{N} p(s_n|z, x)\prod_{n=1}^{N} p(z_n|C). \tag{3}$$

3.2 NEURAL IMPLEMENTATION

Global 3D Representation. The implementation of global representation consists of three steps: (i) scene encoding, (ii) object gathering, and (ii) object encoding. For *scene encoding*, we first perform order-invariant encoding of the context C at image-level (not explicitly aware of objects in it yet) and obtain D-dimensional global scene-encoding e_n for each cell n in $\mathcal{Z}$. We then infer the presence for each cell by sampling a Bernoulli distribution $z_n^{pres} \sim \text{Bern}(f_{\text{nn}}(e_n))$ and, if $z_n^{pres} = 1$, the 3-dimension coordinate for the center position of the object in the 3D space by using a neural network, i.e., $z_n^{pos} \sim \mathcal{N}(f_\mu^{pos}(e_n), f_\sigma^{pos}(e_n))$. The main challenge of our model is in the next step, i.e., learning z_n^{what}. Because we want a viewpoint-invariant 3D representation of a specific object n, it has to *gather the object* from context: detect, segment, and extract only the target object of interest from the context images where multiple objects coexist with occlusion and partial observability and some objects may not be observed due to the viewpoint. Once we crop out and gather these object-wise segments for each existing object, we perform object-encoding to build the 3D-representation corresponding to that specific object.

To implement object gathering, we first notice that, using the 3D-to-2D coordinates projection operation $f_{\text{3D}\to\text{2D}}$, we can compute the center location of an object existing in a context image. Specifically, we can deterministically convert a 3D position z_n^{pos} in the global 3D coordinate system into a 2D position $s_{c,n}^{pos}$ in a 2D image $y_c \in Y_C$, i.e., $s_{c,n}^{pos} = f_{\text{3D}\to\text{2D}}(z_n^{pos}, x_c)$ where $s_{c,n}^{pos}$ refers to the 2D position of object n within context image y_c. For more details on the coordinate projection, refer to Appendix. Besides local center position, we also predict the bounding box scale of the object by $s_{c,n}^{scale} = f_{\text{scale}}(s_{c,n}^{pos}, e_n, x_c)$, thus we know how large is the portion of the object being projected in the context image. We extract the patch $y_{c,n} \subset y_c$ which is a segmentation in y_c corresponding to object n, by using the spatial transformer. Gathering all patches $\{y_{c,n}\}_c$ corresponding to object n, we perform object-encoding to obtain the viewpoint-invariant 3D-representation of appearance z_n^{what} for a specific object n. This is done by an order-invariant object encoder $r_n = f_{\text{order-invar-obj}}(\{y_{c,n}\}_c)$ and then sampling from $p(z_n^{what}|r_n)$. Thus, our encoding process contains two order-invariant encoders, one for global scene-level and another for object-level which is shared across different objects.

Local 2D Representation. For hierarchical object-awareness, the local representation $p(s|z,x) = \prod_{n\in\mathcal{P}} p(s_n^{pres}, s_n^{pose}, s_n^{what}|z,x)$ is implemented as follows. For each global object z_n whose presence is `true`, we predict whether it also exists in the target 2D image y given r_n together with s_n^{pos} and s_n^{scale} obtained in the same way as mentioned above. If it exists (i.e., mapped on the image y), we predict appearance $s_{x,n}^{what}$. Finally, the appearance is obtained by an object-level GQN decoder using ConvDraw, $s_{x,n}^{what} = \text{ConvDRAW}(z_n^{what}, x)$.

Rendering. The main challenge in rendering the local object-oriented representation $s = \{s_n\}_{n=1}^N$ into the image canvas, i.e., $p(y|s)$, is to deal with the occlusion. In ROOTS, this problem can easily be dealt with by noticing that the coordinate projection function $f_{\text{3D}\to\text{2D}}(z_n^{pos}, x)$ actually transforms a 3D coordinate to another 3D coordinate in which the last dimension becomes the distance between the observer's viewpoint x and the object (and thus can be ignored in 2D projection). The idea is that this distance can be interpreted as object depth from viewer's perspective. Then, during reconstruction we can sort objects according to depth and drawing with occlusion becomes easy. To handle the foreground and background segmentation, we learn the background separately in image level and learn an object-wised mask using GQN decoder. Details of implementations are provided in Appendix A.1.

3.3 LEARNING AND INFERENCE

Because of intractable posterior $p(z,s|C,D)$, we train ROOTS using variational inference with posterior approximation $q_\phi(z,s|C,D) = q_\phi(z|C,D)q_\phi(s|z,C,D)$. For continuous latent variables such as the pose and appearance, we can use reparameterization trick (Kingma & Welling, 2013) and for the discrete variables on the presence, a continuous relaxation using Gumbel-Softmax trick (Jang et al., 2016) or other methods based on the REINFORCE algorithm (Williams, 1992; Tucker et al., 2017; Grathwohl et al., 2017) can be used. Implementation of the approximate posterior $q(z|C,D)$ and $q(s|z,C,D)$ is made easier by sharing the parameters of q with that of conditional prior $p(z|C)$: we only need to provide additional data D to the order-invariant encoder. The objective

is to maximize the following evidence lower bound (ELBO): $\mathcal{L}(\theta, \phi; C, D) =$

$$\mathbb{E}_{s,z\sim q_\phi}\left[\log p_\theta(y|x,s) - \mathbb{KL}[q_\phi(s|z,C,D) \parallel p_\theta(s|z,x)]\right] - \mathbb{KL}[q_\phi(z|C,D) \parallel p_\theta(z|C)]. \quad (4)$$

Combining Unconditional Prior. One difficulty in using the conditional prior of GQN is the fact that, because the prior is now learned, we are more limited in reflecting our prior knowledge into the prior distribution than the unconditioned prior. In fact, in our experiments, it turns out that biasing the posteriors of some variables like z^{pres} towards the values of our prior preference is helpful in stabilizing the model. To implement this, in our training, we use the following objective that has some additional KL terms between the posterior and *unconditioned* prior.

$$\begin{aligned}\mathcal{L} = \mathcal{L}(\theta, \phi; C, D) &+ \mathbb{KL}[q_\phi(z^{pos}, s^{scale}|C, D) \parallel \mathcal{N}(0, 1)] \\ &+ \gamma\mathbb{KL}[q_\phi(z^{pres}|C, D) \parallel \text{Geom}(\rho)] + \gamma\mathbb{E}_{z\sim q_\phi}\left[\mathbb{KL}[q_\phi(s^{pres}|z^{pres}, C, D) \parallel \text{Geom}(\rho)]\right]\end{aligned} \quad (5)$$

where the ρ and γ is hyperparameters and we set them to 0.999 and 7 during training. This auxiliary loss can be derived if we replace our conditional prior by the product of expert prior $p(z|C)p(z)$ divided by the posterior $q(z|C)$.

4 Related Works

Neural processes (NP) (Garnelo et al., 2018) and generative query networks (GQN) (Eslami et al., 2018) are important works to our proposed methods. More details on NP and GQN is described in Section 2. Although to our knowledge there is no previous work on unsupervised object-oriented representation learning for 3D, there have been several literature on its 2D problems. The first is the Attend, Infer, Repeat (AIR) model (Eslami et al., 2016). AIR uses spatial transformer (Jaderberg et al., 2015) to crop an object patch and uses an RNN to sequentially generate next object conditioning on the previous objects. In Crawford & Pineau (2019), the authors showed that this RNN-based rollout is inefficient and can degrades performance as the number of objects increases. The authors proposed the SPAIR model inspired by YOLO (Redmon et al., 2016). However, unlike YOLO, SPAIR does not require bounding box labels. Although applying spatially invariant attentions, the main limitation of SPAIR is to infer the latents sequentially. Neural Expectation Maximization (NEM) (Greff et al., 2017) considers the observed image as a pixel-level mixture of K objects, and an image per object is generated and combined according to the mixture probability. A differentiable variation of an EM algorithm is introduced. In Greff et al. (2019), the authors proposed a more efficient version of NEM, called IODINE, using iterative inference Marino et al. (2018), and in MONET (Burgess et al., 2019) an RNN drawing a scene component at each time step is used. Unlike AIR and SPAIR, the representations in NEM, IODINE, and MONET do not explicitly provide natural disentanglement like presence and pose per object.

5 Experiments

We evaluate ROOTS quantitatively and qualitatively. We train ROOTS with the same hyper parameters for all datasets. We first briefly describe the datasets. For more details of the dataset generation and network architecture, refer to Appendix A.3 and Appendix A.1. We use MuJoCo to simulate 3D scenes. Specifically, we generate three different datasets with scenes with 1-3 objects, 2-4 objects and 3-5 objects, respectively. We set the image size to $64 \times 64 \times 3$ pixels. For each object, we randomly choose its position, shape, size and color. Although we put all objects on the floor, we still predict $\{x, y, z\}$ coordinates for z^{pos} because objects have different sizes. We generate 60K different scenes for each dataset and split them into 50k for training, 5k for validation, and 5k for test.

5.1 Qualitative Evaluation

We provide several generation samples comparing our model with GQN for the same scene under the same query cameras. As seen in Figure 2, ROOTS has clearer generations than GQN. To provide a further understanding of the advantages of object-orientated representation, we present decompositions generated from ROOTS. As shown in Figure 3, ROOTS can generate clean background, clear foreground and detailed occlusion mask. We also show that the learned global representation is good enough to recover angle and pose of an object under different viewpoints.

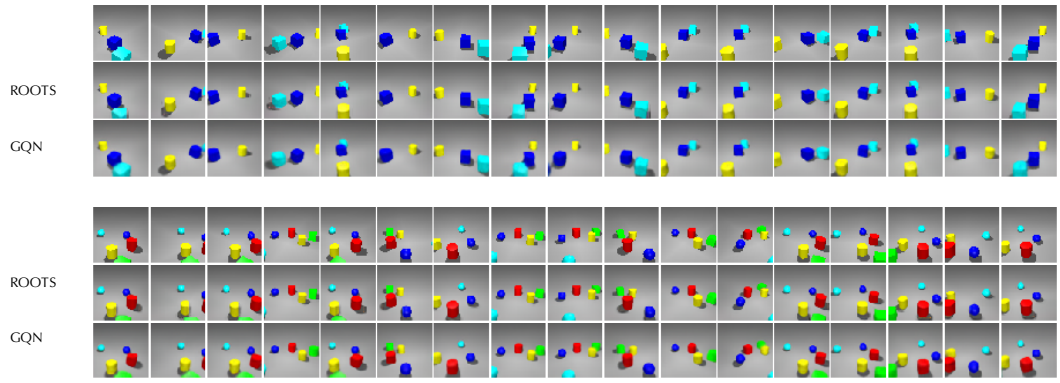

Figure 2: Examples of generations from two different scenes together with ground truth placed in the first row for each scene.

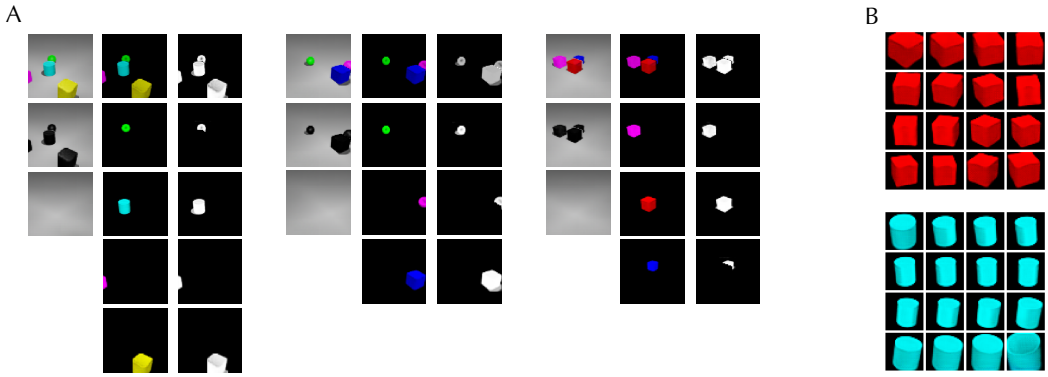

Figure 3: **A**: Generated examples of decomposition for a scene visualized as 2d images captured by cameras. For example, GoodNet segments a scene into foreground and background first and decompose foreground into each individual object further, which are green sphere, blue cylinder, partially observed cube and yellow cube as shown in the first example. We also show the depth mask obtained through object's reconstruction and its distance from camera. **B**: Generations of one object from different cameras.

Object-wise disentanglement: Generations from random viewpoints after changing object positions. In this section and the following sections, we use ROOTS trained on the 2-4 object dataset for visualization unless otherwise stated. To have a better understanding of the structured latent variable learned by ROOTS through context images, we plan to modify the z^{pos} of one object in the scene. As a good disentangled latent representation, this modification should not affect the position, existence and appearance of other objects concurrent in the scene. More importantly, we should be able to change either the x or y coordinate independently and this change should be consistent across viewpoints. To demonstrate this, after changing one dimension of z^{pos} of the red cylinder as shown in Figure 4, we feed this modified $\hat{z}^{pos}$ back to ROOTS. Then, we sample their s^{scale}, s^{pres} and z^{what} in the same way as we do during testing. Generations from 4 different cameras are shown in Figure 4. We can see that ROOTS has a strong ability to learn disentangled representations and occlusions have been handled well due to this advantage.

Compositionality. Another advantage with object-orientated representations is that a new scene can be easily be built with object components, since every scene can be naturally decomposed into several independent objects. Thus, by simple combination, we can build novel scenes. To demonstrate this, we first provide several context images from three different scenes and save the object documentary for each scene, $\{z^{pos}, z^{pres}, z^{what}\}$. Then we swap an object between the first two scenes and add an additional object to the third scene as shown in Figure 5. We can see that the position of each selected object stays the same after this manipulation. Also, by adding one new object, we make a scene with 5 objects, which does not exist in the training dataset.

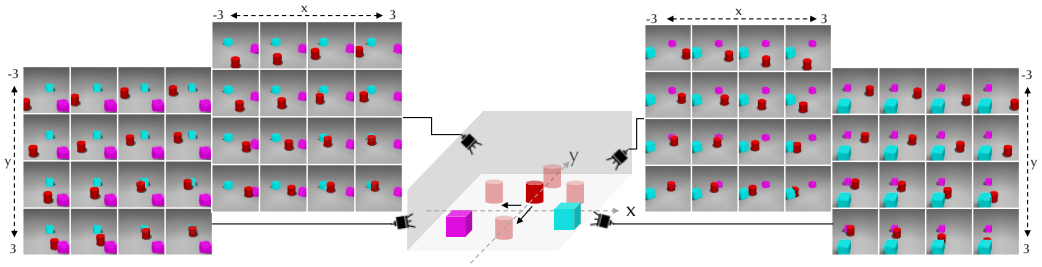

Figure 4: Visualization of changing $z^{pos,x}$ and $z^{pos,y}$ of the red cylinder in a scene through generations from 4 different cameras. We also simulated the 3D scene in the center. Cameras are shown as an example. We put walls for readers to better understand the relative height of cameras. There is no wall in the real dataset.

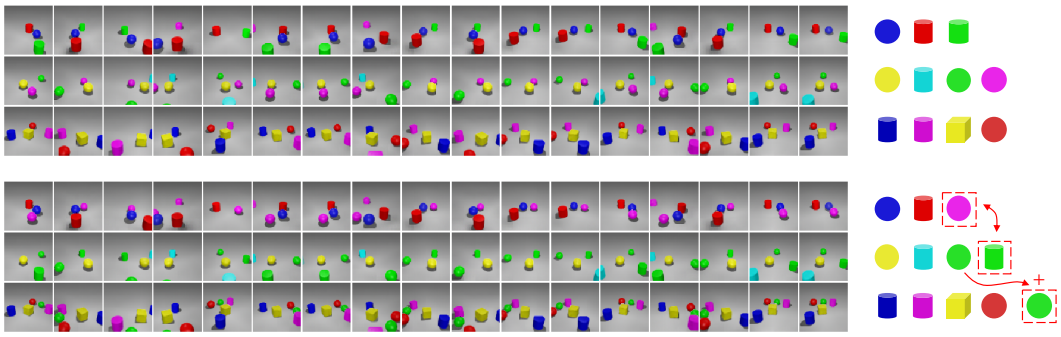

Figure 5: TOP: Original generations from three different scenes, icons on right side highlights objects in the each scene. Each column shows 2D image captured from one camera. Bottom: Manipulated generations from the same three scenes under the same cameras. We switched green cylinder from scene 1 with purple sphere from scene 2 and we copied green sphere from scene 2 and put it in scene 3.

Partial Observations. We know that the successful generations of ROOTS, even with only partial observations provided, is coming from object gathering across viewpoints. Because if an object is missing from one camera, ROOTS can learn it from other cameras. Here, we want to push the partial observation into the extreme case, where one object is totally unobserved by all context cameras. In this case, ROOTS should not be able to correctly predict its existence, just like humans observing the true physical world. To show this, we manually select some images from a scene that have one object missing and the rest serve as targets for ROOTS to generate. We show the results in Figure 6.

5.2 Quantitative Evaluation

NLL and MSE. In this section, we compare the quantitative results of ROOTS and GQN on negative log-likelihood (NLL) and the mean squared error (MSE) on the test set. We approximate NLL using importance sampling with $K = 50$ samples and report the image NLL normalized by the number of pixels in an image in Table 1. Both GQN and ROOTS are trained for 120 epochs on each dataset. We see that, although learning interpretable object-factorized representations, ROOTS achieves NLL comparable to CGQN and MSE which is slightly better than GQN.

	1-3 objects		2-4 objects		3-5 objects	
	NLL	MSE	NLL	MSE	NLL	MSE
ROOTS	0.9199	15.16	0.9205	25.37	0.9213	29.82
GQN	0.9196	15.28	0.9201	26.86	0.9204	32.66

Table 1: Negative log-likelihood and mean squared error

Object Detection Quality. We provide precision and recall results as object detection evaluation of ROOTS. To estimate the true positive prediction, we first filter out predictions that are too far away from any ground truth by applying a radius threshold. The distance is calculated as the euclidean

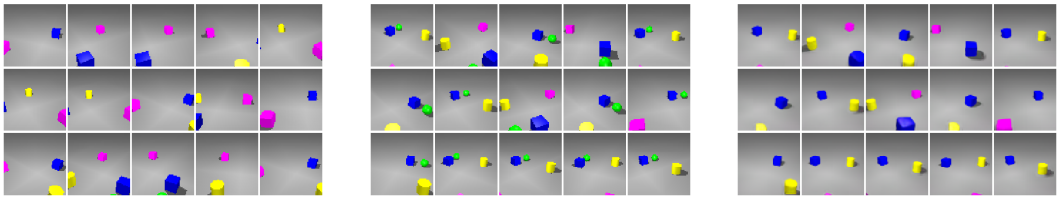

Figure 6: Left: Context images provided to ROOTS, we select images that do not have green sphere projected. Middle: Target images, from which we can see that, there is one green sphere existed. Right: Generations from ROOTS given context images from the left.

distance between two center points. Then we assign the predicted object to the ground truth object to which it has the nearest distance. Multi-association is not allowed in our calculation. We normalize the coordinate value to $[-1, 1]$, thus, the average object size is 0.3. We provide the precision and recall result under different threshold in Table 2. Besides precision and recall, we also provide the counting accuracy of ROOTS.

	1-3 objects			2-4 objects			3-5 objects		
Threshold	0.1	0.15	0.35	0.1	0.15	0.35	0.1	0.15	0.35
Precision	76.82	91.56	98.68	66.76	86.07	98.29	66.64	86.09	96.65
Recall	75.56	90.15	97.40	65.53	84.57	95.93	66.91	86.47	97.10
Count Acc.		93.18			88.16			82.21	

Table 2: Precision and Recall

6 Conclusion

We proposed ROOTS, an unsupervised 3D scene decomposition and 3D object detection network. ROOTS can learn object-oriented interpretable and hierarchical 3D scene representation. We showed the generation, decomposition and detection ability of ROOTS and we also showed that, due to the factorization of structured representation, new scenes can be easily built up by reusing components from 3D scenes. Interesting future directions would be to learn the knowledge of the 3D world in a sequential manner as we humans do and update the inferred knowledge through time.

REFERENCES

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.

Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of AAAI*, 2019.

SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, David P Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pp. 6691–6701, 2017.

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. *arXiv preprint arXiv:1903.00450*, 2019.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pp. 3549–3557, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ananya Kumar, SM Eslami, Danilo J Rezende, Marta Garnelo, Fabio Viola, Edward Lockhart, and Murray Shanahan. Consistent generative query networks. *arXiv preprint arXiv:1807.02033*, 2018.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. *arXiv preprint arXiv:1807.09356*, 2018.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pp. 2627–2636, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

A Appendix

A.1 Implementation Details

Here, we give the details of the implementation of ROOTS. We first outline the generation and inference process for one object, indexed with n. Parallelizing it to multiple objects is straight forward. Then, we will give details of important modules in subsequent sections. Note that $\mathcal{I}(\mathcal{C})$, $\mathcal{I}(\mathcal{D})$ and $\mathcal{I}(\mathcal{S})$ are representing the index for context set, target set and whole sequence set, respectively.

Generation Below, we outline the implementation of the generative model. The SceneRepNet is the module we use for scene level order invariant encoder. The ObjectRepNet is the module we use for object level order invariant encoder. All the DRAW modules has hidden state of channel number 128.

$$r_C \leftarrow \sum_{c \in \mathcal{I}(\mathcal{C})} \text{SceneRepNet}_\theta(x_c, y_c) \quad \text{(Encode scene level context)} \tag{6}$$

$$z_n^{pos} \sim \text{DRAW}_\theta^{pos}(r_C) \quad \text{(Sample 3D position for object } n\text{)} \tag{7}$$

$$z_n^{pres} \sim \text{CONV}_\theta(r_C) \quad \text{(Sample global presence for object } n\text{)} \tag{8}$$

$$s_{i,n}^{scale} \sim \text{DRAW}_\theta^{scale}(r_C, x_i), i \in \mathcal{I}(\mathcal{S}) \quad \text{(Sample size for object } n \text{ under context } c\text{)} \tag{9}$$

$$s_{i,n}^{pos} \leftarrow f_{3D \to 2D}(z_n^{pos}, x_i), i \in \mathcal{I}(\mathcal{S}) \quad \text{(Convert 3D position into pixel location)} \tag{10}$$

$$y_{c,n}^{att} \leftarrow \text{STN}(y_c, [s_{c,n}^{pos}, s_{c,n}^{sacle}]), c \in \mathcal{I}(\mathcal{C}) \quad \text{(Crop object patch)} \tag{11}$$

$$r_{n,C} \leftarrow \sum_{c \in C} \text{ObjectRepNet}(y_{c,n}^{att}, x_c), c \in \mathcal{I}(\mathcal{C}) \quad \text{(Encode object level context)} \tag{12}$$

$$z_n^{what} \sim \text{DRAW}_\theta^{what}(r_{n,C}) \quad \text{(Sample global what for object } n\text{)} \tag{13}$$

$$s_{i,n}^{pres} \sim \text{CONV}_\theta(z_n^{pres}, r_{n,C}, x_i), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Sample local presence for object } n\text{)} \tag{14}$$

$$\hat{y}_{i,n}^{att}, \alpha_{i,n}^{att} \leftarrow \text{Renderer}_\theta(z_n^{what}, x_i, i \in \mathcal{I}(\mathcal{D})) \quad \text{(Decode the object patch and object mask)} \tag{15}$$

$$\hat{y}_{i,n} \leftarrow \text{STN-INV}(\alpha_{i,n}^{att} \times \hat{y}_{i,n}^{att}, [s_{i,n}^{pos}, s_{i,n}^{sacle}]), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Generate objects)} \tag{16}$$

$$\alpha_{i,n} \leftarrow \text{STN-INV}(\alpha_{i,n}^{att}, [s_{i,n}^{pos}, s_{i,n}^{sacle}]), i \in \mathcal{I}(\mathcal{D}) \tag{17}$$

$$\alpha_{i,n}^{occ} \leftarrow depth_n \times \alpha_{i,n} == \min_n(depth_n \times \alpha_{i,n}), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Obtain occlusion mask pixel-wisely)} \tag{18}$$

$$\hat{y}_i^{fg} \leftarrow \sum_n (\alpha_{i,n}^{occ} \times \hat{y}_{i,n} \times s_{i,n}^{pres}), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Generate foreground)} \tag{19}$$

$$\alpha_i^{fg} \leftarrow \sum_n (\alpha_{i,n} \times \alpha_{i,n}^{occ} \times s_{i,n}^{pres}), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Generate foreground mask)} \tag{20}$$

$$z^{bg} \sim \text{DRAW}_\theta^{bg}(r_C) \quad \text{(Sample background)} \tag{21}$$

$$\hat{y}_i^{bg} \leftarrow \text{BgDecoder}(z^{bg}, x_i) \quad \text{(Decode background)} \tag{22}$$

$$\hat{y}_i \leftarrow \hat{y}_i^{fg} + (1 - \alpha_i^{fg}) \times \hat{y}_i^{bg} \quad \text{(Render generations)} \tag{23}$$

Inference The inference model is paralleled with generative model. We only highlight the different part below.

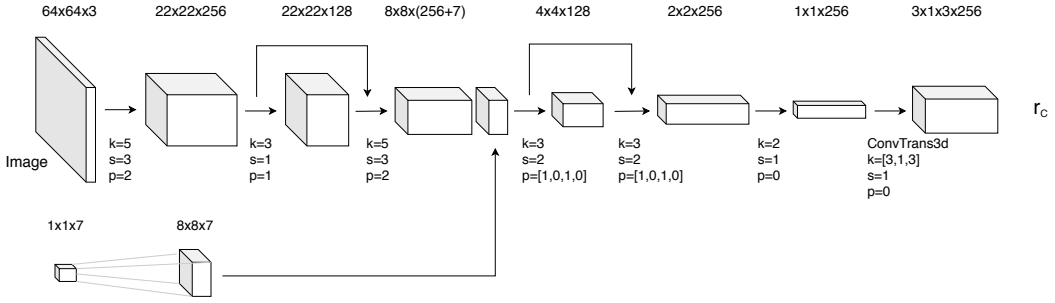

Figure 7: Scene Representation Network

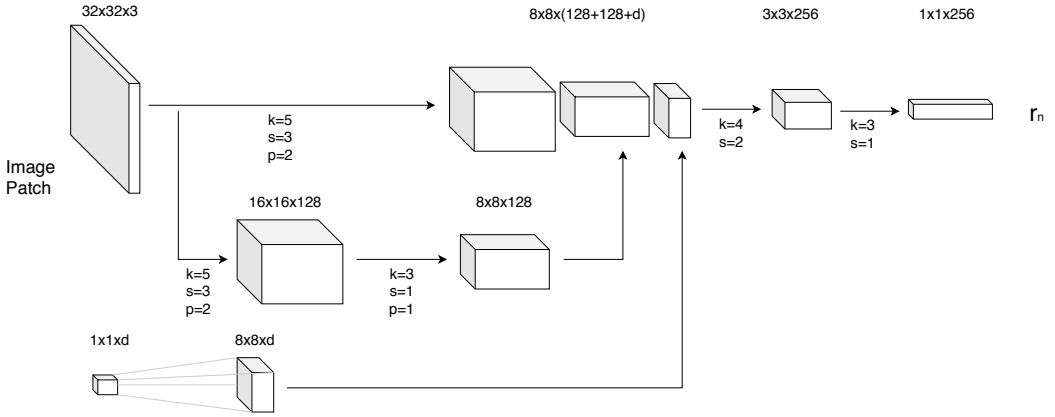

Figure 8: Object Representation Network

$$r_D \leftarrow \sum_{i \in \mathcal{I}(\mathcal{D})} \text{SceneRepNet}_\theta(x_i, y_i) \quad \text{(Encode scene level context)} \tag{24}$$

$$z_n^{pos} \sim \text{DRAW}_\theta^{pos}(r_C, r_D) \quad \text{(Sample 3D position for object } n\text{)} \tag{25}$$

$$z_n^{pres} \sim \text{CONV}_\theta(r_C, r_D) \quad \text{(Sample global presence for object } n\text{)} \tag{26}$$

$$s_{i,n}^{scale} \sim \text{DRAW}_\theta^{scale}(r_C, r_D, x_i), i \in \mathcal{I}(\mathcal{S}) \quad \text{(Sample size for object } n\text{)} \tag{27}$$

$$s_{i,n}^{pos} \leftarrow f_{3D \to 2D}(z_n^{pos}, x_i), i \in \mathcal{I}(\mathcal{S}) \quad \text{(Convert 3D position into pixel location)} \tag{28}$$

$$y_{i,n}^{att} \leftarrow \text{STN}(y_i, [s_{i,n}^{pos}, s_{i,n}^{sacle}]), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Crop object patch)} \tag{29}$$

$$r_{n,D} \leftarrow \sum_{i \in i \in \mathcal{D}} \text{ObjectRepNet}(y_{i,n}^{att}, x_i) \quad \text{(Encode object level context)} \tag{30}$$

$$z_n^{what} \sim \text{DRAW}_\theta^{what}(r_{n,C}, r_{n,D}) \quad \text{(Sample global what for object } n\text{)} \tag{31}$$

$$s_{i,n}^{pres} \sim \text{CONV}_\theta(r_{n,C}, r_{n,D}, x_i), i \in \mathcal{I}(\mathcal{D}) \quad \text{(Sample local presence for object } n\text{)} \tag{32}$$

$$z^{bg} \sim \text{DRAW}_\theta^{bg}(r_C, r_D) \quad \text{(Sample background)} \tag{33}$$

A.2 Details of component modules

Scene Representation Network: The Scene Representation Network is modified bases on Representation Network in GQN.

Object Representation Network: The objcet Representation Network, we design a branch to pass low level feature to provide richer conv-features. The dimension d in the second input, which is concatenation of $\{z^{pres}, s^{scale}, s^{pos}\}$ and parameters of STN, will be clarified in Table 3.

Sufficient Statistics Networks: We list the configuration of all the Sufficient Statistics Networks in Table 3 used during generation. For z_n^{pos}, z_n^{what} and $s_{n,i}^{scale}$, we use auto-regressive scheme to learn the sufficient statistics for each latent variable distributions given hidden state of a ConvLSTM. Specifically, we just apply several convolutional layers, which take the hidden state of ConvLSTM as input, to learn sufficient statistics for target distributions. For z_n^{pres} and $s_{n,i}^{pres}$, we only use regular convolutional layers. Here, we give the details of Sufficient Statistic Networks for each latent variable. In the third column, we give the kernel size of first convolutional layer, the remaining are Conv3D1X1 layers. If the kernel size is 3, we have one zero paddings, stride is 1 to keep the spatial size unchanged. The Draw Rollouts column shows the number of DRAW steps we apply. The Concat column specify how we we use sampled latent value for subsequent process, for example, concatenating z^l,for $l < L$ or only taking z^L.

Table 3: Configuration of Sufficient Statistic Networks

Latent Variable	Channel Numbers	K	Draw Rollouts	Concate
z_n^{pos}	[128, 128, 64, 32, 3]	3	2	last step
z_n^{what}	[128, 128, 64, 32, 4]	1	4	concatenate
$s_{i,n}^{scale}$	[128, 128, 64, 32, 2]	1	4	last step
z_n^{pres}	[256, 256, 128, 64, 1]	3	-	-
$s_{i,n}^{pres}$	[271, 256, 128, 64, 32, 1]	1	-	-
z^{bg}	[128, 4]	3	2	last step

GQN Implementation: We strictly follow the paper for implementation details. The only difference is we enhance the decoder in the renderer by adding two more convolutional layers.

A.3 Dataset Details

The size of objects are randomly chosen from 0.56 to 0.66. Each object is put on the floor of 3D space with range of $[-2, 2]$ along both x-axis and y-axis. We have three different types of object, cube, sphere and cylinder with 6 different colors. For each dataset, we first randomly choose the number objects in a scene, then randomly object type, color and their positions (x and y coordinates). For each scene, we have 30 cameras put at a radius of 3, pointing at a squared area located at the center, thus, the camera would not always look at the center point. The pitch is randomly chosen from $[-\pi/6., -\pi/7.]$ and the yaw is randomly chosen from $[-\pi, \pi]$.