Unsupervised Learning of 3D Structure from Images

Danilo J. Rezende, S. M. Ali Eslami, Shakir Mohamed,
{danilor, aeslami, shakir}@google.com

Peter Battaglia, Max Jaderberg, Nicolas Heess
{peterbattaglia, jaderberg, heess}@google.com

Google DeepMind

Abstract

A key goal of computer vision is to recover the underlying 3D structure from 2D observations of the world. In this paper we learn strong deep generative models of 3D structures, and recover these structures from 3D and 2D images via probabilistic inference. We demonstrate high-quality samples and report log-likelihoods on several datasets, including ShapeNet [2], and establish the first benchmarks in the literature. We also show how these models and their inference networks can be trained end-to-end from 2D images. This demonstrates for the first time the feasibility of learning to infer 3D representations of the world in a purely unsupervised manner.

1 Introduction

We live in a three-dimensional world, yet our observations of it are typically in the form of two-dimensional projections that we capture with our eyes or with cameras. A key goal of computer vision is that of recovering the underlying 3D structure that gives rise to these 2D observations.

The 2D projection of a scene is a complex function of the attributes and positions of the camera, lights and objects that make up the scene. If endowed with 3D understanding, agents can abstract away from this complexity to form stable, disentangled representations, e.g., recognizing that a chair is a chair whether seen from above or from the side, under different lighting conditions, or under partial occlusion. Moreover, such representations would allow agents to determine downstream properties of these elements more easily and with less training, e.g., enabling intuitive physical reasoning about the stability of the chair, planning a path to approach it, or figuring out how best to pick it up or sit on it. Models of 3D representations also have applications in scene completion, denoising, compression and generative virtual reality.

There have been many attempts at performing this kind of reasoning, dating back to the earliest years of the field. Despite this, progress has been slow for several reasons: First, the task is inherently ill-posed. Objects always appear under self-occlusion, and there are an infinite number of 3D structures that could give rise to a particular 2D observation. The natural way to address this problem is by learning statistical models that recognize which 3D structures are likely and which are not. Second, even when endowed with such a statistical model, inference is intractable. This includes the sub-tasks of mapping image pixels to 3D representations, detecting and establishing correspondences between different images of the same structures, and that of handling the multi-modality of the representations in this 3D space. Third, it is unclear how 3D structures are best represented, e.g., via dense volumes of voxels, via a collection of vertices, edges and faces that define a polyhedral mesh, or some other kind of representation. Finally, ground-truth 3D data is difficult and expensive to collect and therefore datasets have so far been relatively limited in size and scope.

In this paper we use deep neural networks to learn generative models of 3D structures, and to recover this structure from 2D images via probabilistic inference. Learning models of 3D structures directly from pixels has been a long-standing research problem and a number of approaches with different levels of underlying assumptions and feature engineering have been proposed. Traditional approaches to vision as inverse graphics [24, 21, 23] and analysis-by-synthesis [28, 34, 19, 35] rely on heavily engineered visual features with which inference of object properties such as shape and pose is substantially simplified. More recent work [19, 5, 4, 37] addresses some of these limitations by learning parts of the encoding-decoding pipeline depicted in figure 2 in separate stages. We discuss other related work in A.1. Unlike existing approaches, our approach is one of the first to learning 3D representations in an unsupervised, end-to-end manner, directly from 2D images.

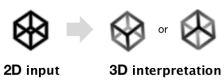

Figure 1: **Motivation:** The 3D representation of a 2D image is ambiguous and multi-modal. We achieve such reasoning by learning a generative model of 3D structures, and recover this structure from 2D images via probabilistic inference.

Our contributions are as follows. (a) We design a strong generative model of 3D structures, defined over the space of volumes and meshes, using ideas from state-of-the-art generative models of images [9]. (b) We show that our models produce high-quality samples, can effectively capture uncertainty and are amenable to probabilistic inference, allowing for applications in 3D generation and simulation. We report log-likelihoods on a dataset of shape primitives, a 3D version of MNIST, and on ShapeNet [2], which to the best of our knowledge, constitutes the first quantitative benchmark for 3D density modeling. (c) We show how complex inference tasks, e.g., that of inferring plausible 3D structures given a 2D image, can be achieved using conditional training of the models. We demonstrate that such models recover 3D representations in one forward pass of a neural network and they accurately capture the multi-modality of the posterior. (d) We explore both volumetric and mesh-based representations of 3D structure. The latter is achieved by flexible inclusion of off-the-shelf renders such as OpenGL [27]. This allows us to build in further knowledge of the rendering process, e.g., how light bounces of surfaces and interacts with its material's attributes. (e) We show how the aforementioned models and inference networks can be trained end-to-end directly from 2D images without any use of ground-truth 3D labels. This demonstrates for the first time the feasibility of learning to infer 3D representations of the world in a purely unsupervised manner.

2 Conditional Generative Models

In this section we develop our framework for learning models of 3D structure from volumetric data or directly from images. We consider conditional latent variable models, structured as in figure 2 (left). Given an observed volume or image $\mathbf{x}$ and a context $\mathbf{c}$, we wish to infer a corresponding 3D representation $\mathbf{h}$ (which can be a volume or a mesh). This is achieved by modelling the latent manifold of object shapes and poses via the low-dimensional codes $\mathbf{z}$. The context is any quantity that is always observed at both train- and test-time, and it conditions all computations of inference and generation (see figure 2, middle). In our experiments, context is either 1) nothing, 2) an object class label, or 3) one or more views of the scene from different cameras.

In order to learn interpretable representations, our models are made to employ a generative process which consists of first generating a 3D representation $\mathbf{h}$ (figure 2, middle) and then projecting to the domain of the observed data (figure 2, right). For instance, the model will first generate a volume or mesh representation of a scene or object and then render it down using a convolutional network or an OpenGL renderer to form a 2D image.

Generative models with latent variables describe probability densities $p(\mathbf{x})$ over datapoints $\mathbf{x}$ implicitly through a marginalization of the set of latent variables $\mathbf{z}$, $p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$. Flexible models can be built by using multiple layers of latent variables, where each layer specifies a conditional distribution parameterized by a deep neural network. Examples of such models include [14, 18, 10, 29]. The marginal likelihood $p(\mathbf{x})$ is intractable and we must resort to approximations. We opt for variational approximations [15], in which we bound the marginal likelihood $p(\mathbf{x})$ by $\mathcal{F} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})]$, where the true posterior distribution is approximated by a parametric family of posteriors $q_\phi(\mathbf{z}|\mathbf{x})$ with parameters ϕ. Learning involves joint optimization of the variational parameters ϕ and model parameters θ. In this framework, we can think of the generative model as a decoder of the latent variables, and the inference network as an encoder of the

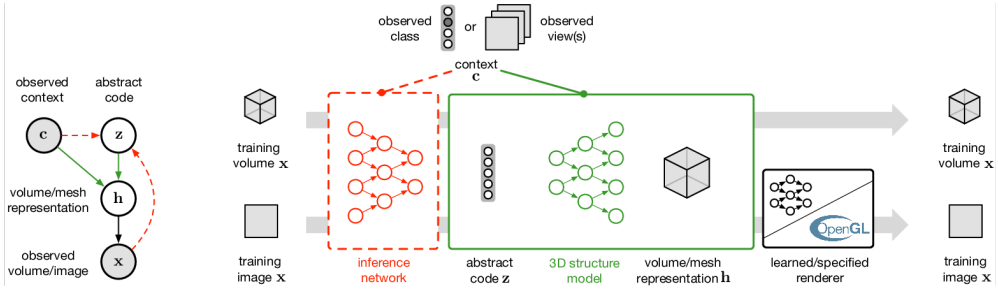

Figure 2: **Proposed framework:** *Left:* Given an observed volume or image $\mathbf{x}$ and contextual information $\mathbf{c}$, we wish to infer a corresponding 3D representation $\mathbf{h}$ (which can be a volume or a mesh). This is achieved by modeling the latent manifold of object shapes via the low-dimensional codes $\mathbf{z}$. In experiments we will consider unconditional models (i.e., no context), as well as models where the context $\mathbf{c}$ is class or one or more 2D views of the scene. *Right:* We train a context-conditional inference network (red) and object model (green). When ground-truth volumes are available, they can be trained directly. When only ground-truth *images* are available, a renderer is required to measure the distance between an inferred 3D representation and the ground-truth image.

observed data into the latent representation. Gradients of $\mathcal{F}$ are estimated using path-wise derivative estimators ('reparameterization trick') [14, 18].

2.1 Architectures

To capture the complex distribution of 3D structures, we apply recent work on *sequential* generative models [9, 13, 8] by extending them to operate on different 3D representations. This family of models generates the observed data over the course of T computational steps. More precisely, these models operate by sequentially transforming independently generated Gaussian latent variables into refinements of a hidden representation $\mathbf{h}$, which we refer to as the 'canvas'. The final configuration of the canvas, $\mathbf{h}_T$, is then transformed into the target data $\mathbf{x}$ (e.g. an image) through a final smooth transformation. In our framework, we refer to the hidden representation $\mathbf{h}_T$ as the '3D representation' since it will have a special form that is amenable to 3D transformations. This generative process is described by the following equations:

$$\text{Latents } \mathbf{z}_t \sim \mathcal{N}(\cdot|\mathbf{0},\mathbf{1}) \quad (1) \qquad \text{3D representation } \mathbf{h}_t = f_{\text{write}}(\mathbf{s}_t, \mathbf{h}_{t-1}; \theta_w) \quad (4)$$

$$\text{Encoding } \mathbf{e}_t = f_{\text{read}}(\mathbf{c}, \mathbf{s}_{t-1}; \theta_r) \quad (2) \qquad \text{2D projection } \hat{\mathbf{x}} = \text{Proj}(\mathbf{h}_T, \mathbf{s}_T; \theta_p) \quad (5)$$

$$\text{Hidden state } \mathbf{s}_t = f_{\text{state}}(\mathbf{s}_{t-1}, \mathbf{z}_t, \mathbf{e}_t; \theta_s) \quad (3) \qquad \text{Observation } \mathbf{x} \sim p(\mathbf{x}|\hat{\mathbf{x}}). \quad (6)$$

Each step generates an independent set of K-dimensional variables $\mathbf{z}_t$ (equation 1). We use a fully connected long short-term memory network (LSTM, [11]) as the transition function $f_{\text{state}}(\mathbf{s}_{t-1}, \mathbf{z}_t, \mathbf{c}; \theta_s)$. The context encoder $f_{\text{read}}(\mathbf{c}, \mathbf{s}_{t-1}; \theta_r)$ is task dependent; we provide further details in section 3.

When using a volumetric latent 3D representation, the representation update function $f_{\text{write}}(\mathbf{s}_t, \mathbf{h}_{t-1}; \theta_w)$ in equation 4 is parameterized by a volumetric spatial transformer (VST, [12]). More precisely, we set $f_{\text{write}}(\mathbf{s}_t, \mathbf{h}_{t-1}; \theta_w) = \text{VST}(g_1(\mathbf{s}_t), g_2(\mathbf{s}_t))$ where g_1 and g_2 are MLPs that take the state $\mathbf{s}_t$ and map it to appropriate sizes. More details about the VST are provided in the appendix A.3. When using a mesh 3D representation f_{write} is a fully-connected MLP.

The function $\text{Proj}(\mathbf{h}_T, \mathbf{s}_T)$ is a projection operator from the model's latent 3D representation $\mathbf{h}_T$ to the training data's domain (which in our experiments is either a volume or an image) and plays the role of a 'renderer'. The conditional density $p(\mathbf{x}|\hat{\mathbf{x}})$ is either a diagonal Gaussian (for real-valued data) or a product of Bernoulli distributions (for binary data). We denote the set of all parameters of this generative model as $\theta = \{\theta_r, \theta_w, \theta_s, \theta_p\}$. Details of the inference model and the variational objective used for their optimization is provided in the appendix A.2.

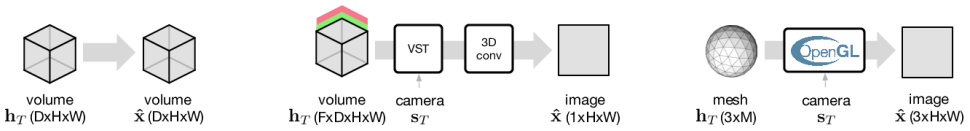

Figure 3: **Projection operators:** These drop-in modules relate a latent 3D representation with the training data. The choice of representation and the type of available training data determine which operator should be used. *Left:* Volume-to-volume projection (no parameters). *Middle:* Volume-to-image neural projection (learnable parameters). *Right:* Mesh-to-image OpenGL projection (no learnable parameters).

Here we discuss the projection operators in more detail. These drop-in modules relate a latent 3D representation with the training data. The choice of representation (volume or mesh) and the type of available training data (3D or 2D) determine which operator is used.

3D $\rightarrow$ 3D projection (identity): In cases where training data is already in the form of volumes (e.g., in medical imagery, volumetrically rendered objects, or videos), we can directly define the likelihood density $p(\mathbf{x}|\hat{\mathbf{x}})$, and the projection operator is simply the identity $\hat{\mathbf{x}} = \mathbf{h}_T$ function (see figure 3 left).

3D $\rightarrow$ 2D neural projection (learned): In most practical applications we only have access to images captured by a camera. Moreover, the camera pose may be unknown or partially known. For these cases, we construct and learn a map from an F-dimensional volume $\mathbf{h}_T$ to the observed 2D images by combining the VST with 3D and 2D convolutions. When multiple views from different positions are simultaneously observed, the projection operator is simply cloned as many times as there are target views. The parameters of the projection operator are trained jointly with the rest of the model. This operator is depicted in figure 3 (middle). For details see appendix A.4.

3D $\rightarrow$ 2D OpenGL projection (fixed): When working with a mesh representation, the projection operator in equation 4 is a complex map from the mesh description $\mathbf{h}$ provided by the generative model to the rendered images $\hat{\mathbf{x}}$. In our experiments we use an off-the-shelf OpenGL renderer and treat it as a black-box with no parameters. This operator is depicted in figure 3 (right).

A key challenge in working with such sophisticated renderers is that of back-propagating errors from the image to the mesh. This requires either a differentiable renderer [23], or resort to gradient estimation techniques such as finite-differences [6] or Monte Carlo estimators [25, 1]. We opt for a scheme based on REINFORCE [33], details of which are provided in appendix A.5.

3 Experiments

We demonstrate the ability of our model to learn and exploit 3D scene representations in five challenging tasks. These tasks establish it as a powerful, robust and scalable model that is able to provide high quality generations of 3D scenes, can robustly be used as a tool for 3D scene completion, can be adapted to provide class-specific or view-specific generations that allow variations in scenes to be explored, can synthesize multiple 2D scenes to form a coherent understanding of a scene, and can operate with complex visual systems such as graphics renderers. We explore four data sets:

Necker cubes The Necker cube is a classical psychological test of the human ability for 3D and spatial reasoning. This is the simplest dataset we use and consists of $40 \times 40 \times 40$ volumes with a $10 \times 10 \times 10$ wire-frame cube drawn at a random orientation at the center of the volume [32].

Primitives The volumetric primitives are of size $30 \times 30 \times 30$. Each volume contains a simple solid geometric primitive (e.g., cube, sphere, pyramid, cylinder, capsule or ellipsoid) that undergoes random translations ($[0, 20]$ pixels) and rotations ($[-\pi, \pi]$ radians).

MNIST3D We extended the MNIST dataset [22] to create a $30 \times 30 \times 30$ volumetric dataset by extruding the MNIST images. The resulting dataset has the same number of images as MNIST. The

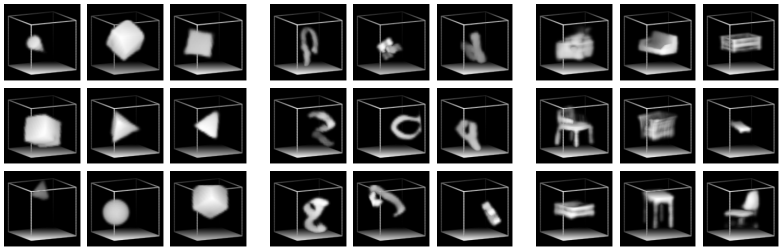

Figure 4: **A generative model of volumes:** For each dataset we display 9 samples from the model. The samples are sharp and capture the multi-modality of the data. *Left:* Primitives (trained with translations and rotations). *Middle:* MNIST3D (translations and rotations). *Right:* ShapeNet (trained with rotations only). Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

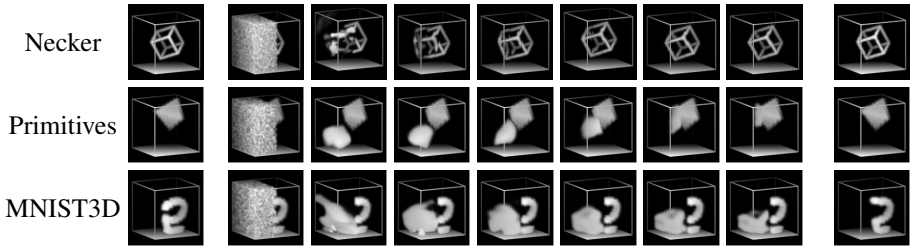

Figure 5: **Probabilistic volume completion (Necker Cube, Primitives, MNIST3D):** *Left:* Full ground-truth volume. *Middle:* First few steps of the MCMC chain completing the missing left half of the data volume. *Right:* 100th iteration of the MCMC chain. Best viewed on a screen. Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

data is then augmented with random translations ($[0, 20]$ pixels) and rotations ($[-\pi, \pi]$ radians) that are procedurally applied during training.

ShapeNet The ShapeNet dataset [2] is a large dataset of 3D meshes of objects. We experiment with a 40-class subset of the dataset, commonly referred to as ShapeNet40. We render each mesh as a binary $30 \times 30 \times 30$ volume.

For all experiments we used LSTMs with 300 hidden neurons and 10 latent variables per generation step. The context encoder $f_c(\mathbf{c}, \mathbf{s}_{t-1})$ was varied for each task. For image inputs we used convolutions and standard spatial transformers, and for volumes we used volumetric convolutions and VSTs. For the class-conditional experiments, the context $\mathbf{c}$ is a one-hot encoding of the class. As meshes are much lower-dimensional than volumes, we set the number of steps to be $T = 1$ when working with this representation. We used the Adam optimizer [17] for all experiments.

3.1 Generating volumes

When ground-truth volumes are available we can directly train the model using the identity projection operator (see section 2.1). We explore the performance of our model by training on several datasets. We show in figure 4 that it can capture rich statistics of shapes, translations and rotations across the datasets. For simpler datasets such as Primitives and MNIST3D (figure 4 left, middle), the model learns to produce very sharp samples. Even for the more complex ShapeNet dataset (figure 4 right) its samples show a large diversity of shapes whilst maintaining fine details.

3.2 Probabilistic volume completion and denoising

We test the ability of the model to impute missing data in 3D volumes. This is a capability that is often needed to remedy sensor defects that result in missing or corrupt regions, (see for instance [36, 5]). For volume completion, we use an unconditional volumetric model and alternate between inference and generation, feeding the result of one into the other. This procedure simulates a Markov chain and samples from the correct distribution, as we show in appendix A.10. We test the model by

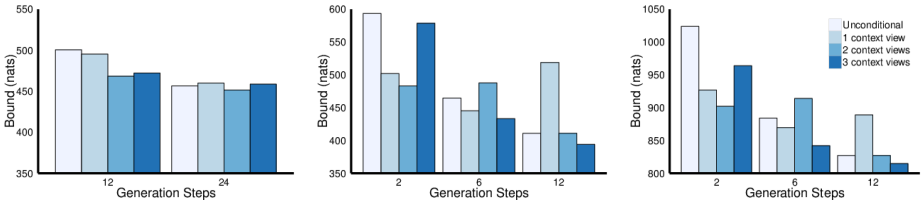

Figure 6: **Quantitative results:** Increasing the number of steps or the number of contextual views both lead to improved log-likelihoods. *Left:* Primitives. *Middle:* MNIST3D. *Right:* ShapeNet.

occluding half of a volume and completing the missing half. Figure 5 demonstrates that our model successfully completes large missing regions with high precision. More examples are shown in the appendix A.7.

3.3 Conditional volume generation

The models can also be trained with context representing the class of the object, allowing for class conditional generation. We train a class-conditional model on ShapeNet and show multiple samples for 10 of the 40 classes in figure 7. The model produces high-quality samples of all classes. We note their sharpness, and that they accurately capture object rotations, and also provide a variety of plausible generations. Samples for all 40 ShapeNet classes are shown in appendix A.8.

We can also form conditional models using a single view of 2D contexts. Our results, shown in figure 8 indicate that the model generates plausible shapes that match the constraints provided by the context and captures the multi-modality of the posterior. For instance, consider figure 8 (right). The model is conditioned on a single view of an object that has a triangular shape. The model's three shown samples have greatly varying shape (e.g., one is a cone and the other a pyramid), whilst maintaining the same triangular projection. More examples of these inferences are shown in the appendix A.9.

3.4 Performance benchmarking

We quantify the performance of the model by computing likelihood scores, varying the number of conditioning views and the number of inference steps in the model. Figure 6 indicates that the number of generation steps is a very important factor for performance. Additional context views generally improve the model's performance but the effect is relatively small. With these experiments we establish the first benchmark of likelihood-bounds on Primitives (unconditional: 500 nats; 3-views: 472 nats), MNIST3D (unconditional: 410 nats; 3-views: 393 nats) and ShapeNet (unconditional: 827 nats; 3-views: 814 nats). All three datasets will be made available upon publication.

3.5 Multi-view training

In most practical applications, ground-truth volumes are not available for training. Instead, data is captured as a collection of images (e.g., from a multi-camera rig or a moving robot). To accommodate this fact, we extend the generative model with a projection operator that maps the internal volumetric representation $\mathbf{h}_T$ to a 2D image $\hat{\mathbf{x}}$. This map imitates a 'camera' in that it first applies an affine transformation to the volumetric representation, and then flattens the result using a convolutional network. The parameters of this projection operator are trained jointly with the rest of the model. Further details are explained in the appendix A.4.

In this experiment we train the model to learn to reproduce an image of the object given one or more views of it from fixed camera locations. It is the model's responsibility to infer the volumetric representation as well as the camera's position relative to the volume. It is clear to see how the model can 'cheat' by generating volumes that lead to good reconstructions but do not capture the underlying 3D structure. We overcome this by reconstructing *multiple* views from the *same* volumetric representation and using pose information to fix a reference frame for the internal volume. This enforces a consistent hidden representation that generalises to new views.

We train a model that conditions on 3 fixed context views to reproduce 10 simultaneous random views of an object. After training, we can sample a 3D representation given the context, and render it from arbitrary camera angles. We show the model's ability to learn to perform this kind of inference

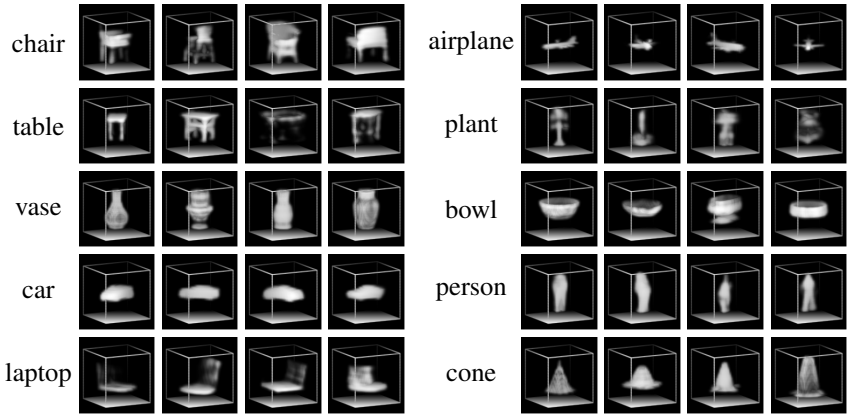

Figure 7: **Class-conditional samples**: Given a one-hot encoding of class as context, the model produces high-quality samples. Notice, for instance, sharpness and variability of generations for 'chair', accurate capture of rotations for 'car', and even identifiable legs for the 'person' class. Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

in figure 9. The resulting network is capable of producing an abstract 3D representation from 2D observations that is amenable to, for instance, arbitrary camera rotations.

3.6 Single-view training

Finally, we consider a mesh-based 3D representation and demonstrate the feasibility of training our models with a fully-fledged, black-box renderer in the loop. Such renderers (e.g. OpenGL) accurately capture the relationship between a 3D representation and its 2D rendering out of the box. This image is a complex function of the objects' colors, materials and textures, positions of lights, and that of other objects. By building this knowledge into the model we give hints for learning and constrain its hidden representation.

We consider again the Primitives dataset, however now we only have access to 2D images of the objects at training time. The primitives are textured with a color on each side (which increases the complexity of the data, but also makes it easier to detect the object's orientation relative to the camera), and are rendered under three lights. We train an unconditional model that given a 2D image, infers the parameters of a 3D mesh and its orientation relative to the camera, such that when textured and rendered reconstructs the image accurately. The inferred mesh is formed by a collection of 162 vertices that can move on fixed lines that spread from the object's center, and is parameterized by the vertices' positions on these lines. The results of these experiments are shown in figure 10. We observe that in addition to reconstructing the images accurately (which implies correct inference of mesh and camera), the model correctly infers the extents of the object not in view, as demonstrated by views of the inferred mesh from unobserved camera angles.

4 Discussion

In this paper we introduced a powerful family of 3D generative models inspired by recent advances in image modeling. We showed that when trained on ground-truth volumes, they can produce high-quality samples that capture the multi-modality of the data. We further showed how common inference tasks, such as that of inferring a posterior over 3D structures given a 2D image, can be performed efficiently via conditional training. We also demonstrated end-to-end training of such models directly from 2D images through the use of differentiable renderers. This demonstrates for the first time the feasibility of learning to infer 3D representations in a purely unsupervised manner.

We experimented with two kinds of 3D representations: volumes and meshes. Volumes are flexible and can capture a diverse range of structures, however they introduce modeling and computational challenges due to their high dimensionality. Conversely, meshes can be much lower dimensional and therefore easier to work with, and they are the data-type of choice for common rendering engines, however standard paramaterizations can be restrictive in the range of shapes they can capture.

It will be of interest to consider other representation types, such as NURBS, or training with a volume-to-mesh conversion algorithm (e.g., marching cubes) in the loop.

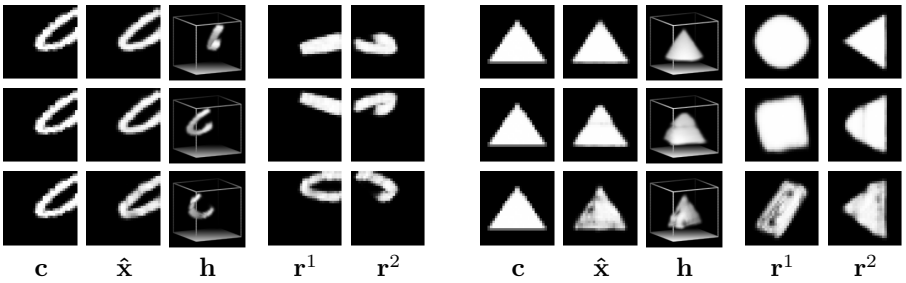

Figure 8: **Recovering 3D structure from 2D images:** The model is trained on volumes, conditioned on $\mathbf{c}$ as context. Each row corresponds to an independent sample $\mathbf{h}$ from the model given $\mathbf{c}$. We display $\hat{\mathbf{x}}$, which is $\mathbf{h}$ viewed from the same angle as $\mathbf{c}$. Columns $\mathbf{r}^1$ and $\mathbf{r}^2$ display the inferred 3D representation $\mathbf{h}$ from different viewpoints. The model generates plausible, but varying, interpretations, capturing the inherent ambiguity of the problem. *Left:* MNIST3D. *Right:* ShapeNet. Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

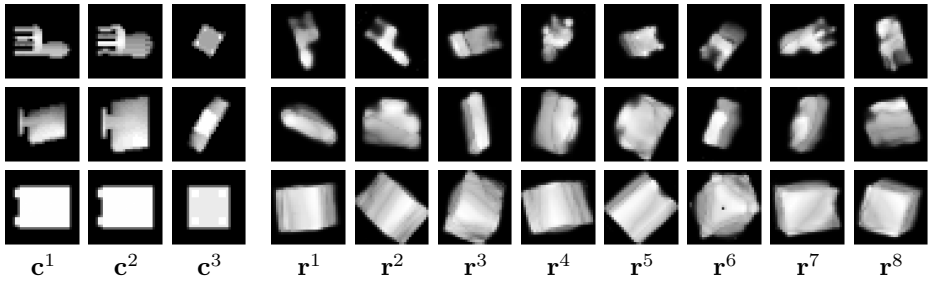

Figure 9: **3D structure from multiple 2D images:** Conditioned on 3 depth images of an object as context, the model is trained to reconstruct depth images of that object from 10 different views. *Left:* Context views. *Right:* Columns $\mathbf{r}^1$ through $\mathbf{r}^8$ display the inferred abstract 3D representation $\mathbf{h}$ rendered from different viewpoints by the learned projection operator. Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

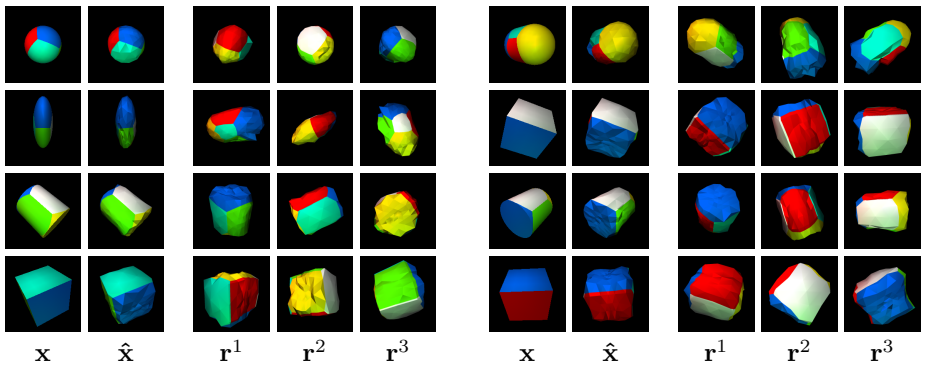

Figure 10: **Unsupervised learning of 3D structure:** The model observes $\mathbf{x}$ and is trained to reconstruct it using a mesh representation and an OpenGL renderer, resulting in $\hat{\mathbf{x}}$. We rotate the camera around the inferred mesh to visualize the model's understanding of 3D shape. We observe that in addition to reconstructing accurately, the model correctly infers the extents of the object not in view, demonstrating true 3D understanding of the scene. Best viewed in color. Videos of these samples can be seen at `https://goo.gl/9hCkxs`.

References

[1] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 30, page 35. ACM, 2011.

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: An unified approach for single and multi-view 3d object reconstruction. *arXiv preprint arXiv:1604.00449*, 2016.

[5] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[6] S. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

[7] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 652–663. ACM, 2004.

[8] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. *arXiv preprint arXiv:1604.08772*, 2016.

[9] K. Gregor, I. Danihelka, A. Graves, D. Jimenez Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

[10] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep autoregressive networks. In *ICML*, 2014.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2008–2016, 2015.

[13] D. Jimenez Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

[14] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[15] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[16] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55, 2012.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[19] T. Kulkarni, I. Yildirim, P. Kohli, W. Freiwald, and J. B. Tenenbaum. Deep generative vision as approximate bayesian computation. In *NIPS 2014 ABC Workshop*, 2014.

[20] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. 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.

[21] T. D. Kulkarni, V. K. Mansinghka, P. Kohli, and J. B. Tenenbaum. Inverse graphics with probabilistic cad models. *arXiv preprint arXiv:1407.1339*, 2014.

[22] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. 1998.

[23] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *Computer Vision–ECCV 2014*, pages 154–169. Springer, 2014.

[24] V. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, pages 1520–1528, 2013.

[25] A. Mnih and D. Jimenez Rezende. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

[26] V. Nair and G. E. Hinton. 3d object recognition with deep belief nets. In *NIPS*, pages 1339–1347, 2009.

[27] OpenGL Architecture Review Board. *OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1*. 1993.

[28] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2719–2726. IEEE, 2012.

[29] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879, 2008.

[30] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, pages 665–673, 2012.

[31] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015.

[32] R. Sundareswara and P. R. Schrater. Perceptual multistability predicted by search model for bayesian decisions. *Journal of Vision*, 8(5):12–12, 2008.

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

[34] D. Wingate, N. Goodman, A. Stuhlmueller, and J. M. Siskind. Nonstandard interpretations of probabilistic programs for efficient inference. In *NIPS*, pages 1152–1160, 2011.

[35] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, pages 127–135, 2015.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[37] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. *arXiv preprint*, May 2016.

A Appendix

A.1 Supplementary related work

Volumetric representations have been explored extensively for the tasks of object classification [31, 26, 30, 36], object reconstruction from images [4], volumetric denoising [36, 4] and density estimation [36]. The model we present in this paper extends ideas from the current state-of-the art in deep generative modelling of images [9, 8, 13] to volumetric data. Since these models operate on smooth internal representations, they can be combined with continuous projection operators more easily than prior work.

On the other hand, mesh representations allow for a more compact, yet still rich, representation space. When combined with OpenGL, we can exploit these representations to more accurately capture the physics of the rendering process. Related work include deformable-parts models [3, 7, 16] and approaches from inverse graphics [35, 20, 6, 23].

A.2 Inference model

We use a structured posterior approximation that has an auto-regressive form, i.e. $q(\mathbf{z}_t|\mathbf{z}_{<t}, \mathbf{x}, \mathbf{c})$. This distribution is parameterized by a deep network:

$$\text{Read Operation} \quad \mathbf{r}_t = f_r(\mathbf{x}, \mathbf{s}_{t-1}; \phi_r) \tag{7}$$

$$\text{Sample} \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}(\mathbf{r}_t,\mathbf{s}_{t-1}, \mathbf{c};\phi_\mu),\sigma(\mathbf{r}_t,\mathbf{s}_{t-1}, \mathbf{c}; \phi_\sigma)) \tag{8}$$

The 'read' function f_r is parametrized in the same way as $f_w(\mathbf{s}_t, \mathbf{h}_{t-1}; \theta_h)$. During inference, the states s_t are computed using the same state transition function as in the generative model. We denote the parameters of the inference model by $\phi = \{\phi_r, \phi_\mu, \phi_\sigma\}$.

The variational loss function associated with this model is given by:

$$\mathcal{F} = -\mathbb{E}_{q(\mathbf{z}_{1,\ldots,T}|\mathbf{x},\mathbf{c})}[\log p_\theta(\mathbf{x}|\mathbf{z}_{1,\ldots,T}, \mathbf{c})] + \sum_{t=1}^{T} \mathrm{KL}[q_\phi(\mathbf{z}_t|\mathbf{z}_{<t}\mathbf{x})\|p(\mathbf{z}_t)], \tag{9}$$

where $\mathbf{z}_{<t}$ indicates the collection of all latent variables from iteration 1 to $t-1$. We can now optimize this objective function for the variational parameters ϕ and the model parameters θ by stochastic gradient descent.

A.3 Volumetric Spatial Transformers

Spatial transformers [12] provide a flexible mechanism for smooth attention and can be easily applied to both 2 and 3 dimensional data. Spatial Transformers process an input image $\mathbf{x}$, using parameters $\mathbf{h}$, and generate an output $\mathrm{ST}(\mathbf{x}, \mathbf{h})$:

$$\mathrm{ST}(\mathbf{x}, \mathbf{h}) = [\kappa_h(\mathbf{h}) \otimes \kappa_w(\mathbf{h})] * \mathbf{x},$$

where κ_h and κ_w are 1-dimensional kernels, $\otimes$ indicates the tensor outer-product of the three kernels and $*$ indicates a convolution. Similarly, Volumetric Spatial Transformers (VST) process an input data volume $\mathbf{x}$, using parameters $\mathbf{h}$, and generate an output $\mathrm{VST}(\mathbf{x}, \mathbf{h})$:

$$\mathrm{VST}(\mathbf{x}, \mathbf{h}) = [\kappa_d(\mathbf{h}) \otimes \kappa_h(\mathbf{h}) \otimes \kappa_w(\mathbf{h})] * \mathbf{x},$$

where κ_d, κ_h and κ_w are 1-dimensional kernels, $\otimes$ indicates the tensor outer-product of the three kernels and $*$ indicates a convolution. The kernels κ_d, κ_h and κ_w used in this paper correspond to a simple affine transformation of a 3-dimensional grid of points that uniformly covers the input image.

A.4 Learnable $3\mathrm{D} \rightarrow 2\mathrm{D}$ projection operators

These projection operators or 'learnable cameras' are built by first applying a affine transformation to the volumetric canvas $\mathbf{c}_T$ using the Spatial Transformer followed a combination on 3D and 2D convolutions as depicted in figure A.4.

A.5 Stochastic Gradient Estimators for Expectations of Black-Box Functions

We employ a multi-sample extension of REINFORCE, inspired by [25, 1]. For each image we sample K realisations of the inferred mesh and compute its corresponding render. The variance of the learning signal for each sample k is reduced by computing a 'baseline' using the $K-1$ remaining samples. See [25] for further details. The estimator is easy to implement and we found this approach to work well in practice even for relatively high-dimensional meshes.

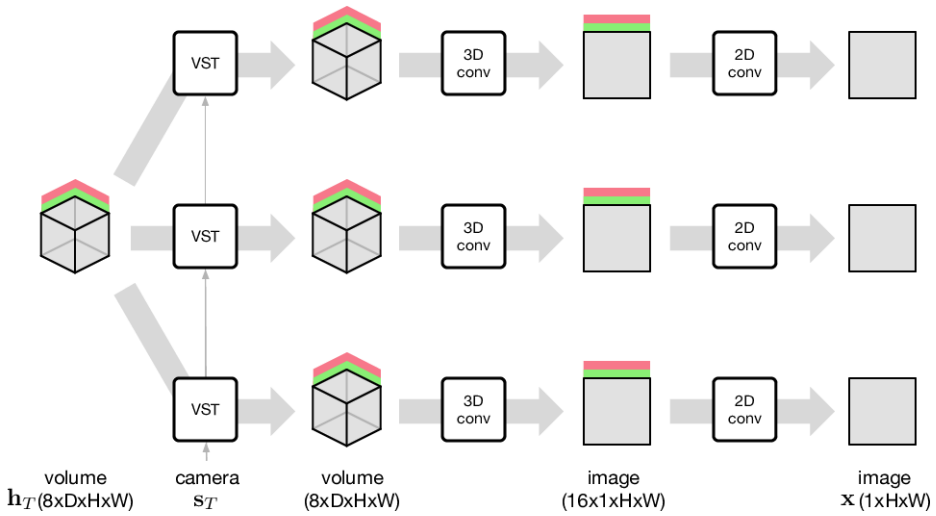

Figure A.1: **Learnable projection operators:** Multiple instances of the projection operator (with shared parameters).

A.6 Unconditional generation

In figures A.2 and A.3 we show further examples of our model's capabilities at unconditional volume generation.

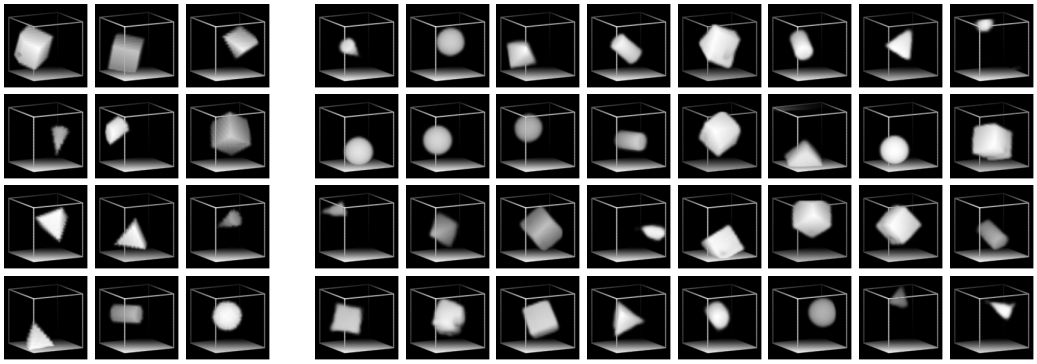

Figure A.2: **A strong generative model of volumes (Primitives):** *Left:* Examples of training data. *Right:* Samples from the model.

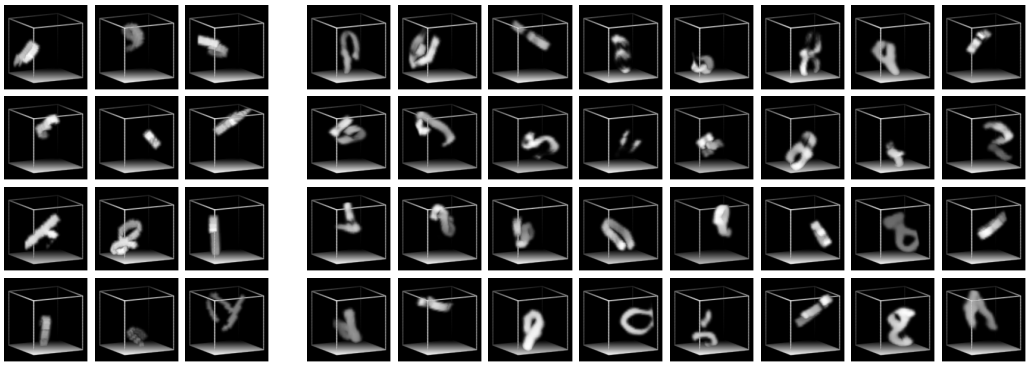

Figure A.3: **A strong generative model of volumes (MNIST3D):** *Left:* Examples of training data. *Right:* Samples from the model.

A.7 Volume completion

In figures A.4, A.5 and A.6 we show the model's capabilities at volume completion.

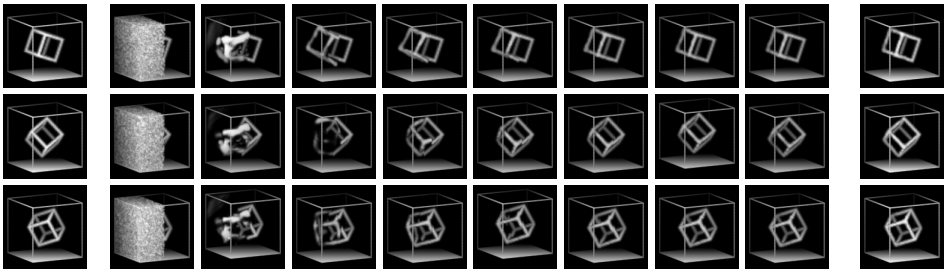

Figure A.4: **Completion using a model of 3D trained on volumes (Necker cube):** *Left:* Full target volume. *Middle:* First 8 steps of the MCMC chain completing the missing left half of the data volume. *Right:* 100th iteration of the MCMC chain.

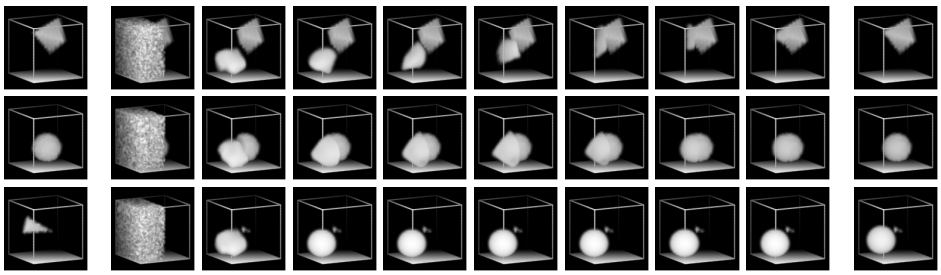

Figure A.5: **Completion using a model of 3D trained on volumes (Primitives):** *Left:* Full target volume. *Middle:* First 8 steps of the MCMC chain completing the missing left half of the data volume. *Right:* 100th iteration of the MCMC chain.

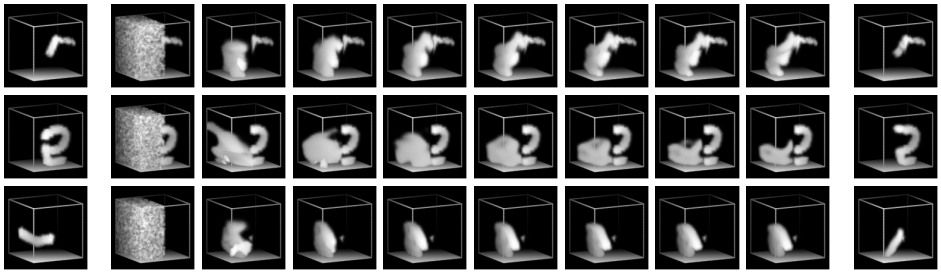

Figure A.6: **Completion using a model of 3D trained on volumes (MNIST3D):** *Left:* Full target volume. *Middle:* First 8 steps of the MCMC chain completing the missing left half of the data volume. *Right:* 100th iteration of the MCMC chain.

A.8 Class-conditional volume generation

In figure A.7 we show samples from a class-conditional volumetric generative model for all 40 ShapeNet classes.

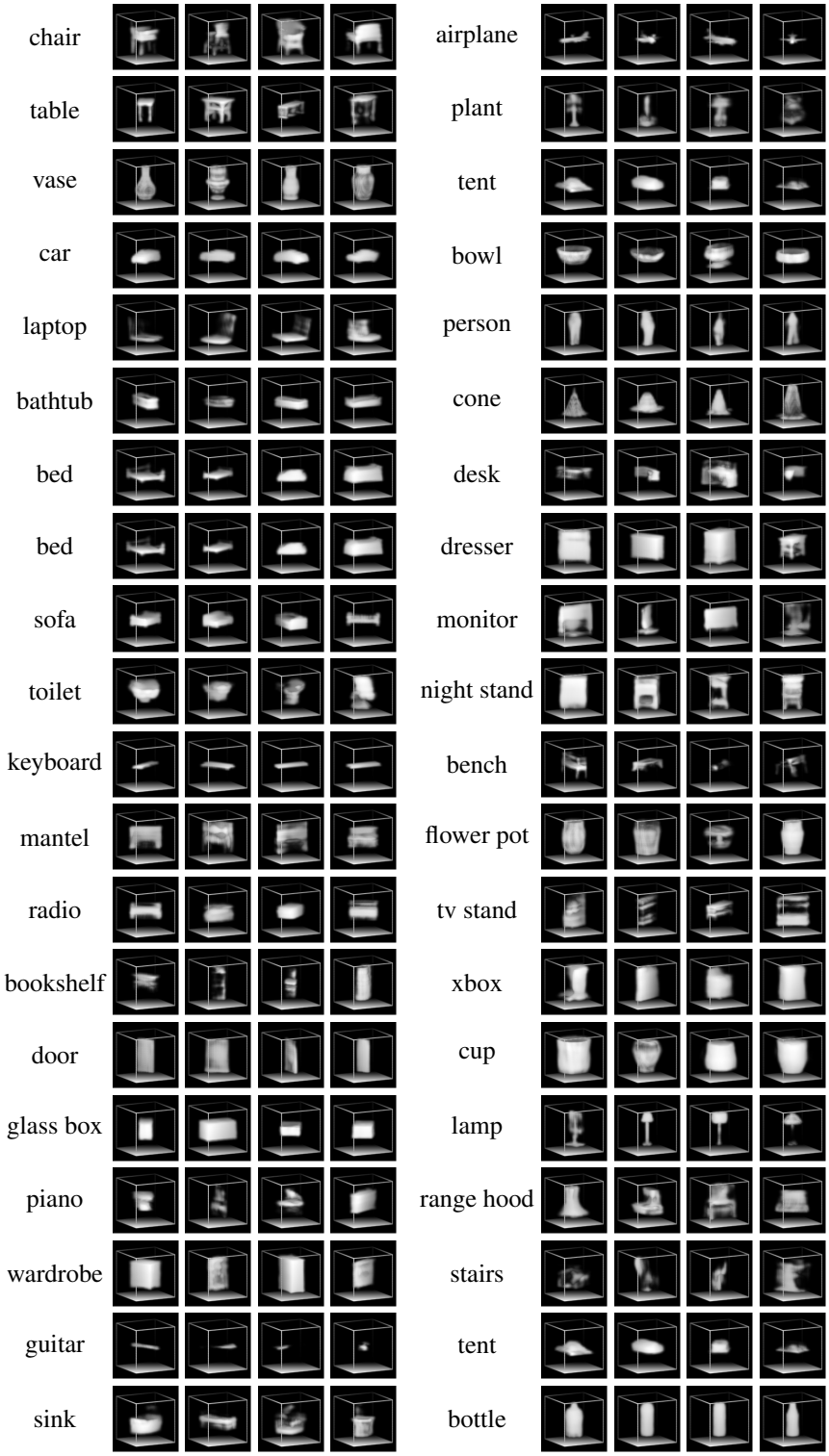

Figure A.7: **Class-Conditional Volumetric Generation (ShapeNet):** All 40 classes.

A.9 View-conditional volume generation

In figures A.8, A.9 and A.10 we show samples from a view-conditional volumetric generative model for Primitives, MNIST3D and ShapeNet respectively.

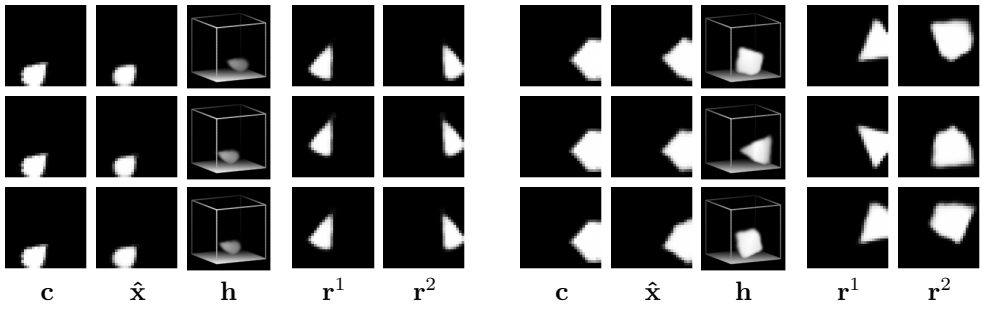

Figure A.8: **Recovering 3D structure from 2D images (Primitives):** The model is trained on volumes, conditioned on $\mathbf{c}$ as context. Each row corresponds to an independent sample $\mathbf{h}$ from the model given $\mathbf{c}$. We display $\hat{\mathbf{x}}$, which is $\mathbf{h}$ viewed from the same angle as $\mathbf{c}$. Columns $\mathbf{r}^1$ and $\mathbf{r}^2$ display the inferred 3D representation $\mathbf{h}$ from different viewpoints. The model generates plausible, but varying, interpretations, capturing the inherent ambiguity of the problem.

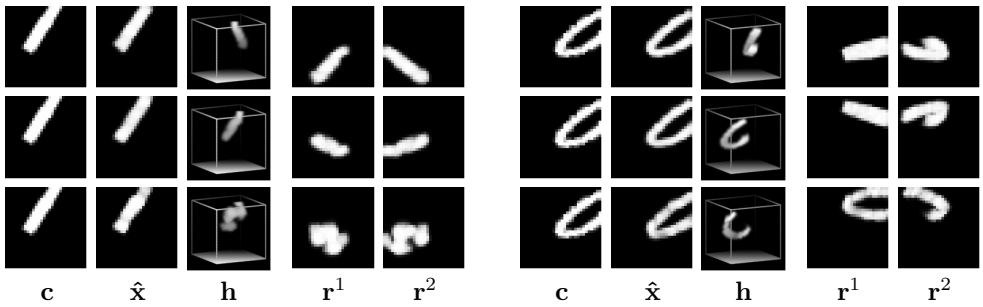

Figure A.9: **Recovering 3D structure from 2D images (MNIST3D):** The model is trained on volumes, conditioned on $\mathbf{c}$ as context. Each row corresponds to an independent sample $\mathbf{h}$ from the model given $\mathbf{c}$. We display $\hat{\mathbf{x}}$, which is $\mathbf{h}$ viewed from the same angle as $\mathbf{c}$. Columns $\mathbf{r}^1$ and $\mathbf{r}^2$ display the inferred 3D representation $\mathbf{h}$ from different viewpoints. The model generates plausible, but varying, interpretations, capturing the inherent ambiguity of the problem.

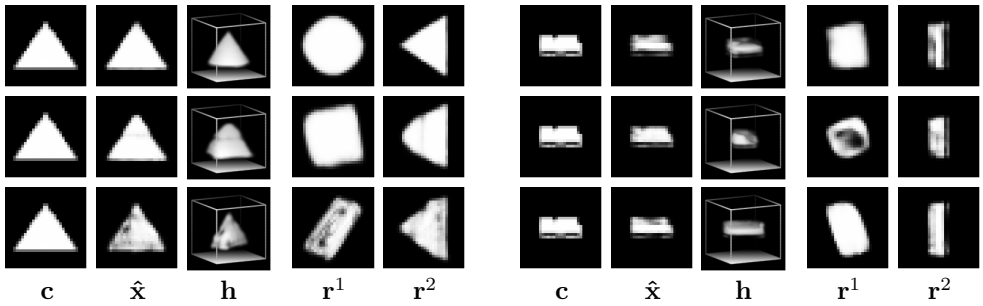

Figure A.10: **Recovering 3D structure from 2D images (ShapeNet):** The model is trained on volumes, conditioned on $\mathbf{c}$ as context. Each row corresponds to an independent sample $\mathbf{h}$ from the model given $\mathbf{c}$. We display $\hat{\mathbf{x}}$, which is $\mathbf{h}$ viewed from the same angle as $\mathbf{c}$. Columns $\mathbf{r}^1$ and $\mathbf{r}^2$ display the inferred 3D representation $\mathbf{h}$ from different viewpoints. The model generates plausible, but varying, interpretations, capturing the inherent ambiguity of the problem.

A.10 Volume completion with MCMC

When only part of the data-vector $\mathbf{x}$ is observed, we can approximately sample the missing part of the volume conditioned on the observed part by building a Markov Chain. We review below the derivations from [14] for completeness. Let $\mathbf{x}_o$ and $\mathbf{x}_u$ be the observed and unobserved parts of $\mathbf{x}$ respectively. All the computations will be conditioned on $\mathbf{x}_o$. The data-completion procedure can be written formally as a Markov chain on the space of missing entries $\mathbf{x}_u$ with transition kernel $\mathcal{K}^q(\mathbf{x}'_u|\mathbf{x}_u, \mathbf{x}_o)$ given by

$$\mathcal{K}^q(\mathbf{x}'_u|\mathbf{x}_u, \mathbf{x}_o) = \iint p(\mathbf{x}'_u, \mathbf{x}'_o|\mathbf{z})q(\mathbf{z}|\mathbf{x})d\mathbf{x}'_o d\mathbf{z}, \tag{10}$$

where $\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_o)$.

The transition kernel (10) is an approximation of the kernel

$$\mathcal{K}(\mathbf{x}'_u|\mathbf{x}_u, \mathbf{x}_o) = \iint p(\mathbf{x}'_u, \mathbf{x}'_o|\mathbf{z})p(\mathbf{z}|\mathbf{x})d\mathbf{x}'_o d\mathbf{z}. \tag{11}$$

The kernel (11) has two important properties: (i) it leaves invariant the conditional $p(\mathbf{x}_u|\mathbf{x}_o)$; (ii) $\mathcal{K}(\mathbf{x}'_u|\mathbf{x}_u, \mathbf{x}_o) > 0 \ \forall \mathbf{x}_o, \mathbf{x}_u, \mathbf{x}'_u$. The property (i) can be derived by applying the kernel (11) to the conditional $p(\mathbf{x}_u|\mathbf{x}_o)$. Property (ii) is an immediate consequence of the smoothness of the model.

From the fundamental theorem for Markov chains, properties (i) and (ii) are sufficient to guarantee that a Markov chain generated by (11) will generate samples from the correct conditional $p(\mathbf{x}_u|\mathbf{x}_o)$.

When using the approximating kernel (10), the stationary distribution of the Markov chain will not be exactly the conditional $p(\mathbf{x}_u|\mathbf{x}_o)$. But the estimator is consistent: if the approximating posterior is sufficiently close to the true posterior, the stationary distribution of the kernel (10) can be shown to be within a certain distance to the correct distribution, as noted in [14].