PIX2SCENE: LEARNING IMPLICIT 3D REPRESENTATIONS FROM IMAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

Modelling 3D scenes from 2D images is a long-standing problem in computer vision with implications in, e.g., simulation and robotics. We propose **pix2scene**, a deep generative-based approach that implicitly models the geometric properties of a scene from images. Our method learns the depth and orientation of scene points visible in images. Our model can then predict the structure of a scene from various, previously unseen view points. It relies on a bi-directional adversarial learning mechanism to generate scene representations from a latent code, inferring the 3D representation of the underlying scene geometry. We showcase a novel differentiable renderer to train the 3D model in an end-to-end fashion, using only images. We demonstrate the generative ability of our model qualitatively on both a custom dataset and on ShapeNet. Finally, we evaluate the effectiveness of the learned 3D scene representation in supporting a 3D spatial reasoning.

1 INTRODUCTION

Understanding and modelling the structure and 3D properties of the real world is a fundamental task with broad applications in engineering, simulation, when training artificial agents and any more general scene inference task. Given that the majority natural scene data is available exclusively in the form of 2D images and videos, the ability to build knowledge about the 3D structure underlying these images would be of great utility to scene understanding-based applications.

A powerful strategy employed to recover 3D structure from images is to model object understanding from a large repository of 3D CAD models (Chang et al., 2015), either by learning the distribution of voxel-based geometric primitives (Wu et al., 2015; 2016c) or by employing retrieval mechanisms (Choy et al., 2016). These strategies can be extended from single objects to entire scenes by further modelling plausible configurations and alignment of the individual objects (again, directly from the image data). On this axis, the state-of-the-art either explicitly models the physical constraints of object-to-scene placement (Huang et al., 2018) or directly infers these relationships (Izadinia et al., 2017). Retrieval-based methods typically lead to scene reconstructions that are realistic and detailed, but limited to only the objects available in the source 3D repository; as such, the resulting 3D scene representations may vary significantly from the actual scene depicted in the input images. The generalization of such strategies to novel or complex scene configurations remains an open problem.

Deep generative models, such as variational autoencoders (VAEs) (Kingma & Welling, 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014), can excel at modeling high-dimensional data, particularly the distribution of natural images (Karras et al., 2018; Zhu et al., 2017; Radford et al., 2015). Recent work using these approaches demonstrate the ability to synthesize novel 3D scenes directly from images: Kulkarni et al. (2015) learn an interpretable and disentangled graphical representation of an underlying image distribution, using VAEs for specific transformations such as out of axis rotation; Rezende et al. (2016) propose a deep generative mechanism to infer the actual 3D representation from 2D images, without any form of direct 3D supervision. Despite promising recent progress, learning high-quality multi-scale representations of arbitrary 3D scene data from images remains a challenge.

We propose **pix2scene**, a deep generative-based approach for modelling the 3D structure of a scene directly from images. We rely on predicting colour and depth as a function of viewpoint in order to infer and distinguish between surface orientation (i.e., normals) and surface depth information.

Thus, our 3D representation of a scene is *implicit*: as the viewpoint changes, the model re-generates appropriate depth and colour information that remains consistent with an underlying (singular) 3D scene structure. This allows us to extrapolate scene information to novel, unobserved viewpoints.

We base our model on an Adversarially Learned Inference (ALI) approach (Dumoulin et al., 2016). ALI extends GAN frameworks by learning an additional inference mechanism: while ALI's encoder maps training samples to latent codes, its decoder plays the role of a more traditional GAN generator, mapping from the latent space to image space. We extend this adversarial framework with 3D priors on the smoothness of depth maps, constraining the generated 3D outputs. This combination of adversarial learning with 3D priors allows us to learn interpretable representations of the 3D scene geometry, directly from 2D images and without any explicit supervision. Our representation is a novel, view-dependent combination of depth maps and (implicit) surface orientations.

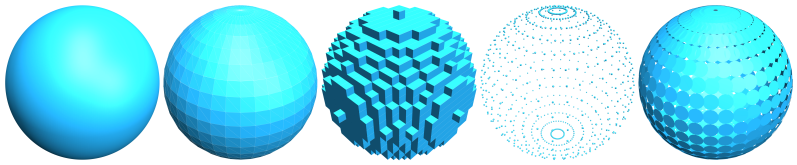
We present several contributions to the field of 3D scene generation. First, unlike prior art, our 3D representation of real-world scenes relies on *view-space surface elements* analogous to the *surfel represnetation* used for rendering in computer graphics (Pfister et al., 2000). These *surf*ace *el*ement*s* comprise the position, orientation/normal, and material reflectance properties of surface patches in a 3D scene. Secondly, we apply a novel adversarial learning-based framework to directly generate these surfel representations, along with an implicit 3D representation of the entire scene, and all this only from input 2D images. Our approach relies fundamentally on a novel surfel-based differentiable 3D renderer to synthesize 2D images from the 3D surfel representations output by our generative model. To the best of our knowledge, this is the first architecture able to learn non-trivial 3D scene models, such as ones composed using objects from ShapeNet (Chang et al., 2015), directly from images. One key novelty of our approach is that we train our model in a completely self-supervised manner, without the need for any depth-based supervision.

2 Related Work

The generation and reconstruction of 3D scenes from images has been studied extensively in the computer vision and graphics communities (Chaudhuri et al., 2011; Chang et al., 2015; Rezende et al., 2016; Soltani et al., 2017). Early (non-parametric) 3D generative models tried to combine object parts in order to form new objects, in a probabilistic manner (Chaudhuri et al., 2011; Kalogerakis et al., 2012). Current methods more often rely on a top-down approach: i.e., shape prior-based methods (Chaudhuri et al., 2011; Choy et al., 2016) learn mappings from 2D observations of single objects to a suitable 3D shape prior, retrieved from a repository of CAD models. These methods can be extended to larger scenes by inferring the 3D layout of the individual objects a priori (Lee et al., 2017) or by considering the layout estimation problem in an end-to-end model (Izadinia et al., 2017; Huang et al., 2018).

The majority of existing methods that estimate 3D structure are based on volumetric or annotation-based representations of a scene (Wu et al., 2016a;b; Soltani et al., 2017). Volumetric (i.e., voxel-based) representations face scalability challenges and, as such, often result in coarse/low-resolution output. Furthermore, volumetric representations somewhat artificially force the generative model into the difficult (and often non-identifiable) task of inferring the unobserved internal volumetric structure of objects (and, more broadly, entire scene elements).

Our work bears some conceptual similarities to Kulkarni et al. (2015) which casts the 3D reconstruction problem as a more traditional inverse graphics task. By using VAEs, they learn a representation of objects that disentangles factors of variations from images (e.g., object pose and configuration). Unlike this approach, ours is fully unsupervised and implicitly generates 3D scene structure from single images. Our mechanism learns a latent representation for the underlying 3D scene, which we later employ to render the scene from novel views and under novel lighting conditions. Similarly, Rezende et al. (2016) infer 3D configurations, adopting a probabilistic inference framework to build a generative model for 3D. Their work combines standard projection mechanisms with gradient estimation methods. Of particular note, their approach requires multiple runs with mechanisms such as REINFORCE (Williams, 1992) in order to infer gradients from the projection layer. This limits the scalability when considering mesh- and/or voxel-based output representations. Our approach, on the other hand, can adapt to arbitrary scene configurations and does not suffer from any such scalability restrictions.



(a) Sphere (b) Mesh (c) Voxels (d) Point cloud (e) Surfels

Figure 1: **3D representations.** Visualization of different computer graphics 3D representations

3 METHOD

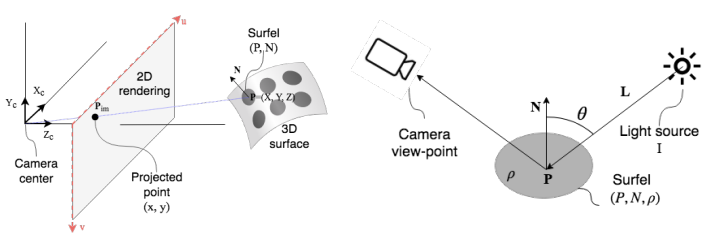
3.1 3D REPRESENTATION: SURFELS

Standard 3D representations such as voxels, meshes or points clouds present different challenges for generative models (Kobbelt & Botsch, 2004). Voxels make the problem tractable but do not scale well with the scene's resolution. The graphical nature of meshes makes it difficult to optimize. As for point clouds, the spatial structure is difficult to use effectively and thus it's also difficult to capture the spatial/rotational invariances (Qi et al., 2017). Accordingly, we propose to model 3D structure with an implicit surface similar to the surfels (Pfister et al., 2000) representation. This representation is very compact: given a renderer's point of view, we can represent only the part of the 3D surface needed by the renderer. Figure 1 illustrates these different representations. For descriptive purpose, surfels are shown as oriented disks, but in general they do not have any shape.

Surfels are represented as a tuple (P, N, ρ), where $P = (p_x, p_y, p_z)$ is its 3D position, $N = (n_x, n_y, n_z)$ is the surface normal vector, and $\rho = (k_r, k_g, k_b)$ is the reflectance of the surface material. Since we are interested in modelling structural properties of the scenes i.e., geometry and depth, we assume that objects in the scene have the uniform material. We represent the surfels in the camera coordinate system. This significantly reduces the number of surfels by considering only the ones that will get projected onto a pixel in the rendered image. Moreover, this allows to reduce the position parameters to only p_z being this the distance along a ray going through the surfel to the center of its pixel.

3.2 DIFFERENTIABLE 3D RENDERER

As our model's critic operates only on image space, we need to project the generated 3D representations back to the 2D space. In a gradient based optimization setting we therefore need a renderer where each stage of the rendering pipeline is differentiable. By using a differentiable renderer as a layer in the neural network, we can backpropagate through the entire network and receive a gradient with respect to the surfels. The rendering process can be partitioned into two stages. The first stage finds the mapping between the surfels and the pixels. In the second stage, the renderer computes the color of each pixel. In terms of automatic differentiation, during backpropagation, the first stage directs the gradients only to the surfels that get projected onto the image, and the second stage is differentiable as long as the shading operations are differentiable.



(a) Projection model (b) Shading model

Figure 2: **Differentiable 3D renderer.** (a) Surfels are defined by its position P and normal N and projected into its corresponding image point P_{im}. (b) The surfel's color depends on the angle θ between the lights I, the surfel's normal N and the reflectance ρ.

The first stage of the rendering involves finding the mapping between the surfels and the pixels. This requires performing the expensive operation of ray object intersection (See figure2a). Therefore our model requires a fast rendering engine as we have to use it in every learning iteration. Conventional

ray tracing algorithms are optimized for generating multiple views from the same scene. However in our setting during learning we render only one image from each scene. Moreover ray tracing algorithms require from representing the full scene which is very inefficient as we only represent the part visible by the camera. To solve these issues, our generator proposes one surfel for each pixel in the camera's coordinate system. Our PyTorch implementation of the differentiable renderer can render a 128×128 surfel-based scene in under 1.4 ms on a mobile NVIDIA GTX 1060 GPU.

The color of a surfel depends on the material's reflectance, its position and orientation, and the ambient and point light source colors. See figure2b. Given a surface point P_i, the color of its corresponding pixel I_{rc} is given by the shading equation:

$$I_{rc} = \rho_i (L_a + \sum_j \frac{1}{k_l \|d_{ij}\| + k_q \|d_{ij}\|^2} L_j \max\left(0, N_i^T d_{ij} / \|d_{ij}\|\right)), \tag{1}$$

where ρ_i is the surface reflectance, L_a is the ambient light's color, L_j is the jth positional light source's color, with $d_{ij} = L_j^{\text{pos}} - P_i$, or the direction vector from the scene point to the point light source, and k_l, k_q being the linear and quadratic attenuation terms respectively.

3.3 PIX2SCENE MODEL

The adversarial training paradigm allows the generator network to capture the underlying target distribution by competing with an adversarial critic network, formulated as a min-max game. Pix2scene employs bi-directional adversarial training to model the distribution of surfels from just 2D images.

3.3.1 BI-DIRECTIONAL ADVERSARIAL TRAINING

ALI (Dumoulin et al., 2016) or Bi-GAN (Donahue et al., 2016) extends the GAN framework by providing a mechanism for learning a representation of the data distribution in an unsupervised manner. Specifically, in addition to the decoder network G_x, ALI provides an encoder G_z which maps data points $\boldsymbol{x}$ to latent representations $\boldsymbol{z}$. In these bi-directional models, critic D discriminates in both the data space ($\boldsymbol{x}$ versus $G_x(\boldsymbol{z})$), and latent space ($\boldsymbol{z}$ versus $G_z(\boldsymbol{x})$), maximizing the adversarial value function over two joint distributions. The final min-max objective can be written as:

$$\min_G \max_D \mathcal{L}(G, D) := \mathbb{E}_{q(\boldsymbol{x})}[\log(D(\boldsymbol{x}, G_z(\boldsymbol{x})))] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D(G_x(\boldsymbol{z}), \boldsymbol{z}))], \tag{2}$$

where $q(\boldsymbol{x})$ and $p(\boldsymbol{z})$ denote encoder and decoder marginal distributions.

3.3.2 MODELLING DEPTH AND CONSTRAINED NORMAL ESTIMATION

Based on the ALI formulation, as depicted in figure 3, our model has an encoder network G_x which captures the distribution over the latent space given an image data point $\boldsymbol{x}$. But it also has a decoder network G_z which maps from a fixed latent distribution $p(\boldsymbol{z})$, a standard normal distribution in our case, to the 3D surfel representation. The resulting surfels are then rendered into a 2D image using our differentiable renderer. This image is then given as input to the critic to distinguish from the real image data. Note that the input to the critic comes from the joint space of data with its corresponding latent code, as in ALI.

A straightforward way to model the decoder network could be to learn a conditional distribution to produce the surfel's depth (p_z) and normal (N). But this could lead to inconsistencies between the local shape and the surface normal. For instance, the decoder can fake an RGB image of a 3D shape simply by changing the normals while keeping the depth fixed. Furthermore, real-world surfaces can be considered to be planar in a small neighborhood. Since we only consider surfels that are visible to the camera, the surface normal is constrainted to be in the direction of the camera. Considering the camera to be looking along the $-z$ direction, the estimated normal has the constraint $n_z > 0$. Therefore, the local surface normal is estimated by solving the following problem for every surfel,

$$\begin{aligned} \|N^T \nabla P\| &= 0 \\ \text{subject to, } \|N\| &= 1 \text{ and } n_z > 0, \end{aligned} \tag{3}$$

where the spatial gradient ∇P is computed using the 8-neighbour points, and P is the position of the surfels in the camera coordinate system obtained by backprojecting the generated depth along rays.

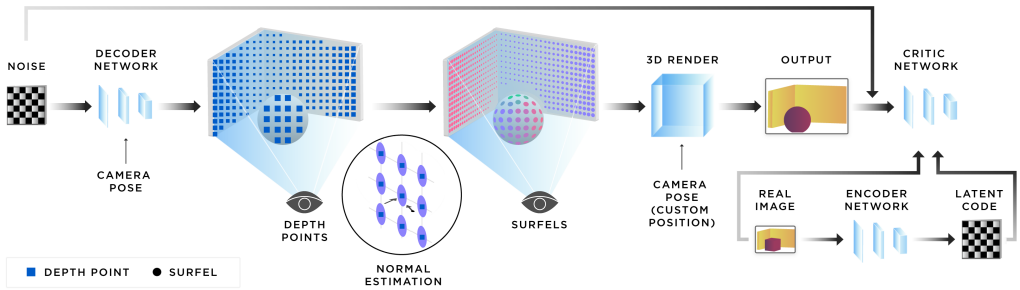


Figure 3: **Pix2scene model.** Pix2scene generates realistic 3D views of scenes by training on 2D images alone. Its decoder generates the surfels depth p_z from a noise vector $\boldsymbol{z}$ conditioned on the camera pose. The surfels normal is estimated from its predicted depth. The surfels are then rendered into a 2D image and together with image samples from the target distribution are fed to the critic.

This approach makes faking 3D a difficult job for the model by just changing the normals. If the depth is incorrect, the normal-estimator outputs a random set of normals corresponding to the incorrect depth, and these random set of normals would result in an incorrect RGB image, which would in-turn get penalized by the critic. The decoder and the encoder networks would therefore be smoothly updated to obtain realistic depths in order to fool the critic network.

3.3.3 Model Training

The Wasserstein-GAN (Arjovsky et al., 2017) formalism provides stable training dynamics using the 1-Wasserstien distance between the distributions. In a high dimensional setting the standard GAN KL divergence could turn out to be too strong a comparison metric. We adopt the gradient penalty setup as proposed in Gulrajani et al. (2017) for more robust training, however, we modify the formulation to take into account the bidirectional training.

Architectures of our networks and training hyper parameters are explained in detail in appendix A. We deal with $128 \times 128 \times 3$ resolution for all our experiments. We found Conditional Normaliation (Dumoulin et al., 2016; Perez et al., 2017) was a better alternate for conditioning the view point than concatenation, which is more traditionally employed for conditional GANs (Mirza & Osindero, 2014). In our training, the affine parameters of the Batch-Normaliation layers (Ioffe & Szegedy, 2015) are replaced by learned representations based on the view point. In order to make sure the reconstructions from the model do not deviate too much from its inputs we also minimize the reconstruction error. Li et al. (2017) argue that minimizing the reconstruction error fixes the identifiability issues with adversarial bi-directional models by minimizing the conditional entropy.

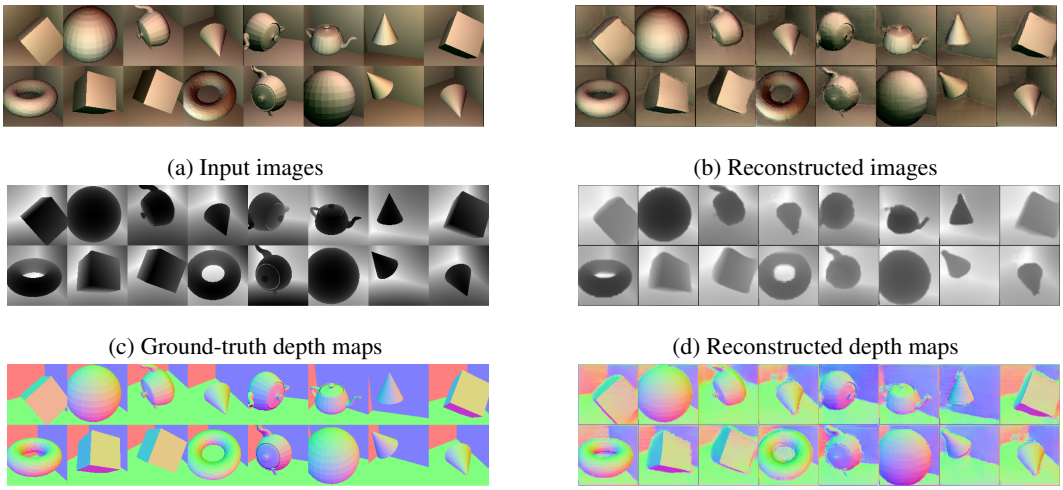
4 Experiments

Experiments were carefully designed to exhibit the model's understanding of the underlying 3D structure of the 2D input data without any form of annotations.

4.1 Implicit 3D Scene Reconstruction

To assess the ability of the network to learn basic 3D shapes from 2D images we trained our model on an extensive set of simulated scenes. The training data consists of rooms with different objects (e.g., teapot, torus, box, sphere, cone) placed in random orientations and positions. Each training image is rendered from a random view sampled uniformly on the positive octant of a sphere containing the room. Technically, the probability of seeing the same configuration of a scene from two different views is near zero. Figure 4 shows the input data and its corresponding reconstructions, along with its recovered depth and normal maps. The depth map is encoded in such a way that the darkest points are closer to the camera. The normal map colors correspond to the cardinal directions (red/green/blue for x/y/z axis respectively). Figure 5 showcases similar results on a dataset comprising of rotating

boxes in a room. Section B in the appendix provides quantitative evaluation of reconstructions from the model using surfel based Hausdorff distance measurement.



(a) Input images (b) Reconstructed images

(c) Ground-truth depth maps (d) Reconstructed depth maps

(e) Ground-truth normal maps (f) Reconstructed normal maps

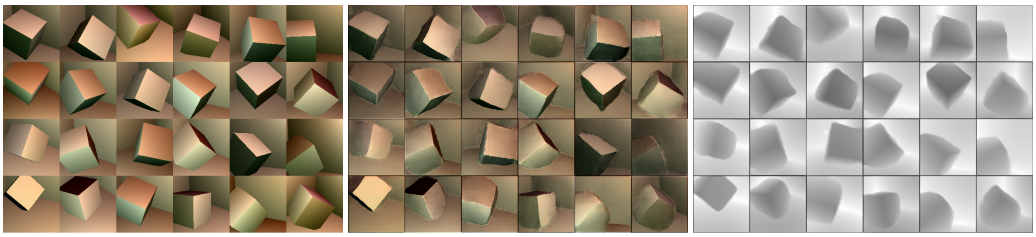
Figure 4: **Scene reconstruction. Left:** Input images of rotated objects into a room with its depth and normal groundtruth maps. **Right:** pix2scene reconstructions with its depth and normal maps.

4.2 Implicit 3D Scene Generation

The ShapeNet dataset (Chang et al., 2015) contains 220,000 3D models classified into 3,135 categories. We used ShapeNetCore which is a subset of ShapeNet that contains 57,386 models from 55 different object categories. To asses the ability of the network to generate new 3D shapes we trained it on 2D images of various objects from 6 classes of the ShapeNet dataset: bowls, bottles, mugs, cans, caps and bags. We have created simple scenes by adding those objects inside a room such that the objects translate randomly relative to the background in each training instance. As shown in figure 6, our model is able to generate correct 3D interpretations of the world that are also visually appealing and original. We also trained our model conditionally by giving the class label of the ShapeNet object to the decoder and critic. During inference we conditioned the generator on the object class we wanted to sample from. Figure 7 shows the results of conditioning the generator on different classes.

4.3 Representation Analysis

In this section we show the richness of pix2scene learned representation. Its learned representations capture the true underlying scene configuration by teasing apart view-based information from scene geometry.



(a) Input images (b) Reconstructed images (c) Reconstructed depth maps

Figure 5: **Scene reconstruction.** (a) Input images of rotated cubes into a room. (b) pix2scene reconstructions with its (c) associated depth maps.

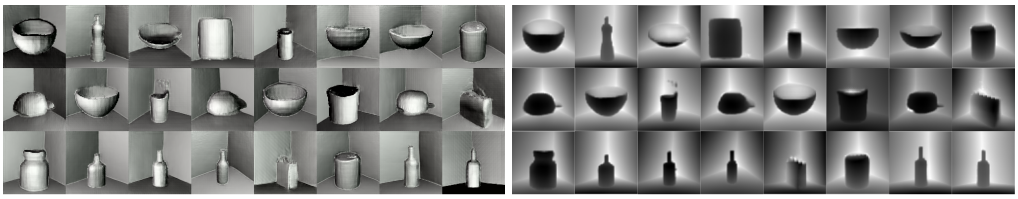


Figure 6: **Unconditional scene generation.** Generated samples from pix2scene model trained on rooms with ShapeNet objects (i.e., bowls, bottles, mugs, cans, caps and bags) located at random positions. **Left:** shaded images; **Right:** depth maps

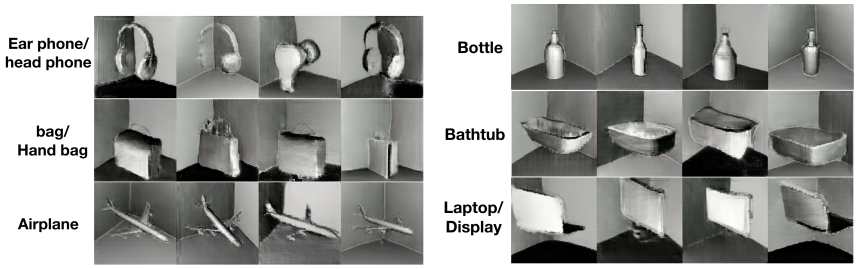


Figure 7: **Conditional scene generation.** Class conditioned generated samples for ShapeNet dataset.

In order to explore the manifold of the learned representations we select two images $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ from the held out data, then we linearly interpolate between its encodings $\boldsymbol{z_1}$ and $\boldsymbol{z_2}$ and decode those intermediary points into its corresponding images. Figure 8 shows this for two different setups. In each case we can see that our representations capture the geometrical aspects of the scene.

In order to showcase the ability of pix2scene to generate views from unobserved angles of a scene during inference time, we infer the latent code $\boldsymbol{z}$ of a chosen image $\boldsymbol{x}$ and then we decode and render different views while rotating the camera around the scene. Figure 9 shows how pix2scene correctly infers the extents of the scene not in view in a consistent manner, demonstrating true 3D understanding of the scene. We show this phenomenon again on different training setups from primitives to complex scene configurations

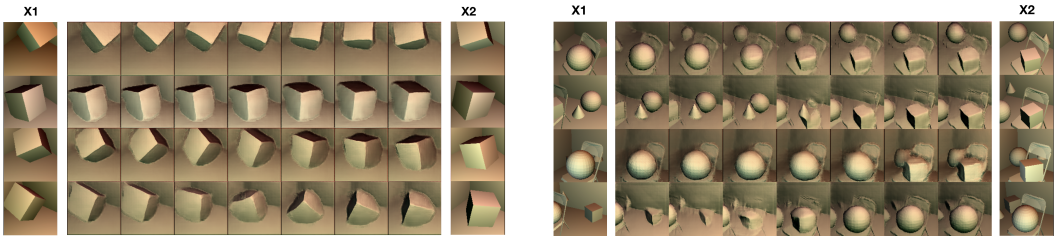


Figure 8: **Representation analysis.** Exploration of the learned manifold of 3D representations. Generated interpolations (middle columns) between two images $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (first and last columns).

4.4 3D-IQ Test Task

We have designed a quantitative evaluation for 3D reconstruction which we refer to as the 3D-IQ test task (3D-IQTT). Some IQ tests, like the Woodcock-Johnson test (Woodcock et al., 2001), employ a mental rotation task where the test taker is presented with a Tetris-like shape as a reference and has to find the rotated version of that shape among a few distractors. Analogously, we generated a test where each IQ question instance consists of a reference image, as well as 3 other images, one of which is a randomly rotated version of the reference (see figure 11 for an example). The training set is formed by 200k questions where only a few are labelled with the information about the correct answer (i.e., either 5% (10k) or 0.5% (1k) of the total training data). The validation and test sets each

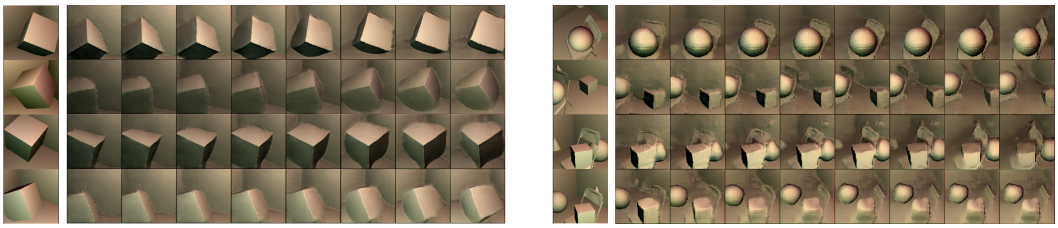


Figure 9: **Representation analysis.** Given a scene (first column), we rotate the camera around to visualize the models understanding of 3D shape. As shown, the model correctly infers the unobserved geometry of the objects, demonstrating true 3D understanding of the scene. Figure 13 in the Appendix depicts the normal maps for the box setup. Videos of these reconstructions can be seen at https://bit.ly/2zADuqG.

contain 100K labelled questions. To verify that our approach is able to learn accurate embeddings of these shapes we first assessed the reconstruction of these shapes qualitatively as shown in figure 10.

4.4.1 SEMI-SUPERVISED CLASSIFICATION ON THE 3D-IQ TEST TASK

The goal of the 3D-IQTT is to quantify how well we can leverage our generative model to perform 3D spatial reasoning on the test set by using the unlabeled training data to extract the necessary 3D information. The evaluation metric is the percentage of correctly answered *questions*.

For training pix2scene in a semi-supervised setting, in addition to the unlabeled data, we also used the labelled data. The training with the unlabeled samples differs from the approach described for previous experiments as we do not assume that we have the camera position available. Thus, part of the latent vector $\boldsymbol{z}$ encodes the actual 3D object (denoted as $\boldsymbol{z}_{scene}$) and the remainder encodes the rotation (denoted as $\boldsymbol{z}_{view}$). For the supervised training three additional loss terms were added: (a) a loss that enforces the object component to be the same for both the reference object and the correct answer, (b) a loss that maximizes the distance between geometric component of reference and the distractors, and (c) a loss that minimizes the mutual information between $\boldsymbol{z}_{scene}$ and $\boldsymbol{z}_{view}$. Losses (a) and (b) are contained in equation 4 where d_i denotes the distractors, $\boldsymbol{x}_{ref}$ is the reference and $\boldsymbol{x}_{ans}$ the correct answer. We interleaved the unsupervised training every 100 steps with complete iterations over the supervised data.

$$\mathcal{L}_\theta(\boldsymbol{x}_{ref}, \boldsymbol{x}_{d_1}, \boldsymbol{x}_{d_2}, \boldsymbol{x}_{ans}) = \frac{1}{2} D_\theta(\boldsymbol{x}_{ref}, \boldsymbol{x}_{ans}) - \frac{1}{2} \sum_{i=1}^{2} D_\theta(\boldsymbol{x}_{ref}, \boldsymbol{x}_{d_i})$$
$$\text{where } D_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2) = (||\boldsymbol{z}^{\boldsymbol{x}_1}_{scene} - \boldsymbol{z}^{\boldsymbol{x}_2}_{scene}||_2)^2 \text{ and } \boldsymbol{z}^{\boldsymbol{x}} = Encoder_\theta(\boldsymbol{x}) \tag{4}$$

The loss (c) is implemented via MINE (Belghazi et al., 2018). The strategy of MINE is to parameterize a variational formulation of the mutual information in terms of a neural network:

(a) Input images (b) Reconstructed images (c) Reconstructed depth map

Figure 10: **3D IQ test task.** Pix2scene reconstructions of the 3D-IQTT shapes.

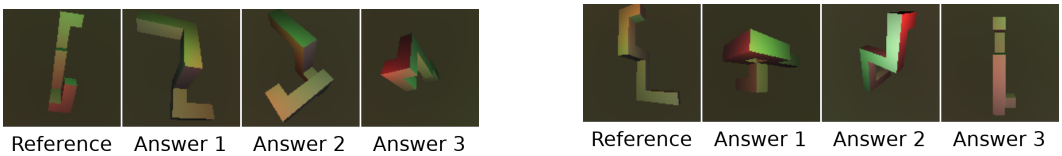


Figure 11: **Sample questions from the 3D-IQ test task.** For this "mental rotation" task, a set of reference images and 3 possible answers are presented. The goal is to find the rotated version of the reference 3D model. To solve this task, the human or the model has to infer the 3D shape of the reference from the 2D image and compare that to the inferred 3D shapes of the answers. The correct answers to these two examples are in the footnote.

Labeled Samples	CNN	Siamese CNN	Pix2Scene **(Ours)**
1,000	0.3392	0.3701	0.5789
10,000	0.3649	0.3752	0.6165

Table 1: **3D-IQTT quantitative results.** The test accuracy of the 3D-IQ test task show that the CNN baselines struggle to solve this task Pix2scene is able to understand the underlying 3D structure of the images and solve the task

$$I_\Theta(z_s, z_v) = \sup_{\theta \in \Theta} \mathbb{E}_{\mathbb{P}_{z_s z_v}}[T_\theta] - \log(\mathbb{E}_{\mathbb{P}_{z_s} \otimes \mathbb{P}_{z_v}}[e^{T_\theta}]). \tag{5}$$

This objective is optimized in an adversarial paradigm where T, the statistics network, plays the role of the critic and is fed with samples from the joint and marginal distribution. We added this loss to our pix2scene objective to minimize the mutual information estimate in both unsupervised and supervised training iterations. Once the model is trained, we answer 3D-IQTT questions, by infering the latent 3D representation for each of the 4 images and we select the answer closest to the reference image as measured by L2 distance. We compared our model to two different baselines. The first one is composed of 4 ResNet-50 modules (He et al., 2016) with shared weights followed by 3 fully-connected layers. We trained this CNN only on the labeled samples. Our second baseline has a similar architecture as the previous one but the fully-connected layers were removed. Instead of the supervised loss provided in the form of correct answers, it is trained on the contrastive loss (Koch et al., 2015). This loss reduces the feature distance between the references and correct answers and maximizes the feature distance between the references and incorrect answers. A more detailed description of the networks and contrastive loss function can be found in the appendix A.1.

Table 1 shows the 3D-IQTT results for our method and the baselines. The baselines were not able to interpret the underlying 3D structure of the data and its results are only slightly better than a random guess. The subpar performance of the Siamese CNN might be in part because the contrastive loss rewards similarities in pixel space and has no notion of 3D similarity. However, pix2scene achieved almost double these scores (0.6165) by leveraging the 3D knowledge of the objects for solving the task.

5 Discussion

In this paper we proposed a generative approach to learn 3D structural properties from still images. To achieve this, we leveraged the generative capability and efficient inference pathway provided by the adversarial bi-directional models. We showcased the model's ability to learn true representations of the 3D scenes from image pixels through a variety of experiments from 3D primitives to challenging scenes. In turn, this learned representation allowed us to synthesize novel scene configurations. We showcased our generative capabilities on the ShapeNet dataset. Lastly, we provided quantitative evidence that support our argument by introducing a novel IQ-task in a semi-supervised setup.

[1] three [2] two

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *International Conference on Machine Learning (ICML)*, 2017.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Devon Hjelm, and Aaron Courville. Mutual information neural estimation. In *International Conference on Machine Learning*, 2018.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv*, 2015.

Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. In *ACM SIGGRAPH*, 2011.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *ArXiv*, 2016.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pp. 1735–1742. IEEE, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image. *arXiv*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Hamid Izadinia, Qi Shan, and Steven Seitz. IM2CAD. *arXiv*, 2017.

Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions in Graphics*, 31(4):55:1–55:11, 2012.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *International Conference on Learning Representations (ICLR)*, 2018.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

Tejas D Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B Tenenbaum. Deep Convolutional Inverse Graphics Network. *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. *CoRR*, abs/1703.06241, 2017.

Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *Advances in Neural Information Processing Systems*, pp. 5501–5509, 2017.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernockỳ. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, 2011.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *Arxiv*, 2014.

Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single RGB image. *CVPR*, 2018.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, 2000.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2015.

Danilo J Rezende, S M Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised Learning of 3D Structure from Images. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Abdel Aziz Taha and Allan Hanbury. An efficient algorithm for calculating the exact hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

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

Richard Woodcock, Nancy Mather, and Kevin McGrew. Woodcock johnson iii - tests of cognitive skills. *Riverside Pub*, 2001.

Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William Freeman, and Joshua s Tenenbaum. Marrnet: 3d shape reconstruction via 2.5d sketches. *ArXiv*, 2016a.

Jiajun Wu, Tianfan Xue, Joseph Lim, Yuandong Tian, Joshua Tenenbaum, Antonio Torralba, and William s Freeman. Single image 3d interpreter network. *ArXiv*, 2016b.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. *Advances in Neural Information Processing Systems (NIPS)*, 2016c.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

A ARCHITECTURE

Pix2scene network architecture is composed by an encoder (See table 2), a decoder (See table 3), and a critic (See table 4). Specifically, the decoder architecture is similar to the generator in DCGAN (Radford et al., 2015) but with LeakyReLU (Mikolov et al., 2011) as activation function and batch-normalization (Ioffe & Szegedy, 2015). Also, we adjusted its depth and width to accommodate the high resolution images accordingly. In order to condition the camera position on the z variable, we use conditional normalization in the alternate layers of the decoder. We train our model for 60K iterations with a batchsize of 6 with images of resolution $128 \times 128 \times 3$.

Encoder					
Layer	Number of outputs	Kernel size	Stride	BatchNorm	Activation
Input $[x, c]$	$128 \times 128 \times 3$				
Convolution	$64 \times 64 \times 85$	4×4	2	Yes	LeakyReLU
Convolution	$32 \times 32 \times 170$	4×4	2	Yes	LeakyReLU
Convolution	$16 \times 16 \times 340$	4×4	2	Yes	LeakyReLU
Convolution	$8 \times 8 \times 680$	4×4	2	Yes	LeakyReLU
Convolution	$4 \times 4 \times 1360$	4×4	2	No	LeakyReLU
Convolution	$1 \times 1 \times 1$	4×4	1	No	

Table 2: **Pix2scene encoder architecture**

Decoder					
Layer	Number of outputs	Kernel size	Stride	BatchNorm	Activation
Input $[x, c]$	131×1				
Convolution	$4 \times 4 \times 1344$	4×4	1	Yes	LeakyReLU
Convolution	$8 \times 8 \times 627$	4×4	2	Yes	LeakyReLU
Convolution	$16 \times 16 \times 336$	4×4	2	Yes	LeakyReLU
Convolution	$32 \times 32 \times 168$	4×4	2	Yes	LeakyReLU
Convolution	$64 \times 64 \times 84$	4×4	2	Yes	LeakyReLU
Convolution	$128 \times 128 \times nCh$	4×4	2	Yes	

Table 3: **Pix2scene decoder architecture.**

Critic					
Layer	Number of outputs	Kernel size	Stride	BatchNorm	Activation
Input $[x, c]$	$128 \times 128 \times 6$				
Convolution	$64 \times 64 \times 85$	4×4	2	No	LeakyReLU
Convolution	$32 \times 32 \times 170$	4×4	2	No	LeakyReLU
Convolution	$16 \times 16 \times 340$	4×4	2	No	LeakyReLU
Convolution	$8 \times 8 \times 680$	4×4	2	No	LeakyReLU
Convolution + [z]	$4 \times 4 \times 1360$	4×4	2	No	LeakyReLU
Convolution	$1 \times 1 \times 1$	4×4	1	No	

Table 4: **Pix2scene critic architecture.** Conditional version takes image, latent code z and camera position c.

A.1 ARCHITECTURE FOR 3D IQTT EVALUATIONS

Pixel2Scene architecture remains similar to the ones in previous sections but with higher capacity on decoder and critic as this task is more challenging and complex. The more important difference is that for those experiments we do not condition the networks with the camera pose to be fair with the baselines. In addition to the three networks we have a statistics network (see table 5) that estimates and minimizes the mutual information between the two set of dimensions in the latent code using MINE(Belghazi et al., 2018). Out of 128 dimensions for z we use first 118 dimensions for represent scene-based information and rest to encode view based info.

Statistics Network					
Layer	Number of outputs	Kernel size	Stride	BatchNorm	Activation
Input $[z[:118], z[118:]]$	$1 \times 1 \times 128$				
Convolution	$1 \times 1 \times 256$	1×1	1	No	ELU
Convolution	$1 \times 1 \times 512$	1×1	1	No	ELU
Convolution	$1 \times 1 \times 1$	1×1	2	No	None

Table 5: **Pix2scene statistics network architecture.**

The architecture of the baseline networks is shown in figure 12. The contrastive loss using for training this baselines is shown in figure 6.

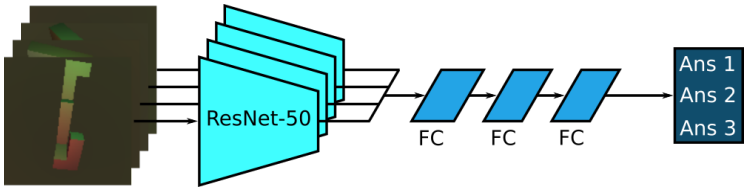


Figure 12: **3D-IQTT baseline architecture.** The ResNet-50 all share the same weights and were slightly modified to support our image size. "FC" stands for fully-connected layer and the hidden node sizes are 2048, 512, and 256 respectively. The output of the network is encoded as one-hot vector.

The contrastive loss from equation 6 is applied to the 2048 features that are generated by each ResNet block. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the input images, y is either 0 (if the inputs are supposed to be the same) or 1 (if the images are supposed to be different), G_θ is each ResNet block, parameterized by θ, and m is the margin, which we set to 2.0. The loss function is from Hadsell et al. (2006) but used slightly differently.

$$\begin{aligned} \mathcal{L}_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2, y) &= (1-y)\frac{1}{2}(D_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2))^2 + (y)\frac{1}{2}(max(0, m - D_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2)))^2 \\ D_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2) &= ||G_\theta(\boldsymbol{x}_1) - G_\theta(\boldsymbol{x}_2)||_2 \end{aligned} \tag{6}$$

B EVALUATION OF 3D RECONSTRUCTIONS

For evaluating 3D reconstructions, we use the Hausdorff distance (Taha & Hanbury, 2015) as a measure of similarity between two shapes as in Niu et al. (2018). Given two point sets, A and B, the Hausdorff distance is, $\max\left\{\max D_H^+(A,B), \max D_H^+(B,A)\right\}$, where D_H^+ is an asymmetric Hausdorff distance between two point sets. E.g., $\max D_H^+(A,B) = \max D(a,B)$, for all $a \in A$, or the largest Euclidean distance $D(\cdot)$, from a set of points in A to B, and a similar definition for the reverse case $\max D_H^+(B,A)$.

We used Hausdorff distance to compare the surfels positions. Specifically, we measured our model's 3D reconstruction's correspondence with the inputs of a specific camera position on a held-out test data. Table 6 shows a quantitative evaluation of the forward and reverse Hausdorff distances on three different datasets. The table also depicts mean squared error of the generated depth map with respect to the input depth map.

Metric	box in a room (rand rotation)	objects in a room (rand rotation)	box in a room (rand translation)
Hausdorff-F	0.102	0.125	0.087
Hausdorff-R	0.183	0.191	0.093
MSE depth map	0.022	0.038	0.032

Table 6: **Scene reconstruction results.** Hausdorff metric on 3D surfel positions and MSE on the depth maps.

C NORMAL MAPS FOR VIEW VARIATIONS

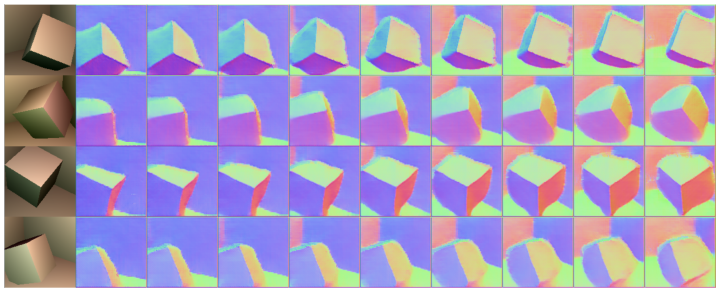


Figure 13: **Normal views.** Fixing representation and and smoothly changing camera view. For each row, the first column is the real input and other columns are the extrapolated views of that image from different views. Figure shows the corresponding normal maps of box inside a room setup in 9 for the extrapolations of box data (left) to stress the consistency.