Transformable Bottleneck Networks

Kyle Olszewski[13*], Sergey Tulyakov[2], Oliver Woodford[2], Hao Li[134], and Linjie Luo[5*]
[1]University of Southern California, [2]Snap Inc., [3]USC ICT, [4]Pinscreen Inc., [5]ByteDance Inc.

Abstract

We propose a novel approach to performing fine-grained 3D manipulation of image content via a convolutional neural network, which we call the Transformable Bottleneck Network (TBN). It applies given spatial transformations directly to a volumetric bottleneck within our encoder-bottleneck-decoder architecture. Multi-view supervision encourages the network to learn to spatially disentangle the feature space within the bottleneck. The resulting spatial structure can be manipulated with arbitrary spatial transformations. We demonstrate the efficacy of TBNs for novel view synthesis, achieving state-of-the-art results on a challenging benchmark. We demonstrate that the bottlenecks produced by networks trained for this task contain meaningful spatial structure that allows us to intuitively perform a variety of image manipulations in 3D, well beyond the rigid transformations seen during training. These manipulations include non-uniform scaling, non-rigid warping, and combining content from different images. Finally, we extract explicit 3D structure from the bottleneck, performing impressive 3D reconstruction from a single input image. [1]

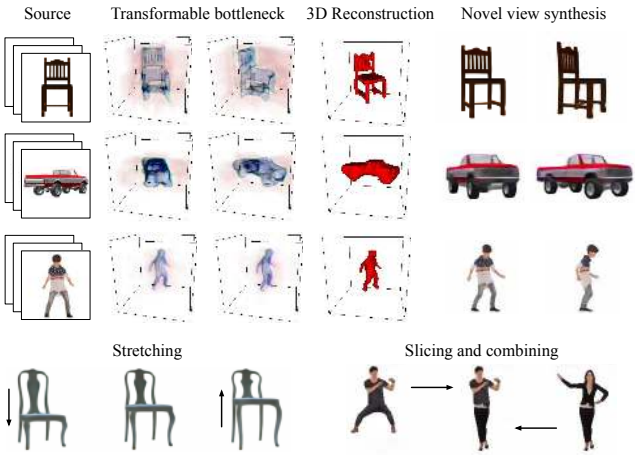


Figure 1: **Applications of TBNs**. A Transformable Bottleneck Network uses one or more images (*column 1*; here, 4 randomly sampled views) to encode volumetric bottlenecks (*columns 2 & 3*), which are explicitly transformed into and aggregated in an output view coordinate frame. Transformed bottlenecks are then decoded to synthesize state-of-the-art novel views (*columns 5 & 6*), as well as reconstruct 3D geometry (*column 4*). Fine-grained and non-rigid transformations, as well as combinations, can be applied in 3D, allowing creative manipulations (*bottom row*) that were never used during training. Images shown are samples of real results.

1. Introduction

Inferring and manipulating the 3D structure of an image is a challenging task, but one that enables many exciting applications. By rigidly transforming this structure, one can synthesize novel views of the content. More general transformations can be used to perform tasks such as warping or exaggerating features of an object, or fusing components of different objects. Convolutional Neural Networks (CNNs) have shown impressive results on various 2D image synthesis and manipulation tasks, but specifying such fine-grained and varied 3D manipulations of the image content, while achieving high-quality synthesis results, remains difficult.

Several approaches to providing transformation parameters as an input to, and applying such transformations within, a network have been explored. A common approach is to pass spatial transformation parameters as an explicit input vector to the network [31], optionally with a decoder trained to perform a specific set of transformations [3, 30]. Other approaches include altering the input by augmenting it with auxiliary channels defining the desired spatial transformation [21], or constructing a renderable representation that is spatially transformed prior to rendering [19, 32].

We propose a novel approach: directly applying the spatial transformations to a volumetric bottleneck within an encoder-bottleneck-decoder network architecture. We call these *Transformable Bottleneck Networks* (TBNs). The network learns that these 3D transformations correspond to transformations between source and target images.

There are several advantages to this approach. Firstly, supervising on multi-view datasets encourages the network to infer spatial structure—it learns to spatially disentangle the feature space within the bottleneck. Consequently, even when training a network using only *rigid* transformations corresponding to viewpoint changes, we can manipulate the network output at test time with *arbitrary* spatial transformations (see Figs. 1 & 6). The operations enabled by these

*This work was performed while the author was at Snap Inc.

[1]Code and data for this project are available on our website: https://github.com/kyleolsz/TB-Networks

transformations thus include not only rotation and translation, but also effects such as non-uniform 3D scaling and global or local non-rigid warping. Additionally, bottleneck representations of multiple inputs can be transformed into, and combined in, the same coordinate frame, allowing them to be aggregated naturally in feature space. This can resolve ambiguities present in a representation from a single image. While similar to ideas in Spatial Transformer Networks (STN) [14, 18] and a 3D reconstruction method [27] deriving from it, a key distinction of our approach is that the spatial transformations are input to our network, as opposed to inferred by the network. It is precisely this difference that enables TBNs to make such diverse manipulations.

We highlight the power of this approach by applying it to novel view synthesis (NVS). NVS is a challenging task, requiring non-trivial 3D understanding from one or more images in order to predict corresponding images from new viewpoints. This allows us to demonstrate both the ability of a TBN to naturally spatially disentangle features within a 3D bottleneck volume, and the benefits that this confers. We compare to leading NVS methods [30, 42, 29, 23], on images from the ShapeNet dataset [1], and attain state-of-the-art results on both L_1 and SSIM metrics (see Table 1, and Figs. 1 & 3a). We present additional qualitative results on a synthetic human performance dataset. We also train a simple voxel occupancy classifier on image segmentations (*i.e.* without 3D supervision), and use it to demonstrate accurate 3D reconstructions from a single image. Finally, we provide qualitative examples of how this bottleneck structure allows us to perform realistic, varied and creative image manipulation in 3D (Figs. 1 & 6).

In summary, the main contributions of this work are:

- A novel, transformable bottleneck framework that allows CNNs to perform spatial transformations for highly controllable image synthesis.
- A state-of-the-art NVS system using TBNs.
- A method for extracting high-quality 3D structure from this bottleneck, constructed from a single image.
- The ability to perform realistic, varied and creative 3D image manipulation.

2. Related work

We now review works related to the TBN, in the areas of image and novel view synthesis, and volumetric reconstruction[2] and rendering.

2.1. Image and novel view synthesis

Many exciting advances in image synthesis and manipulation have emerged recently that enable the application of specific styles or attributes. Early approaches generated natural images using samples from a chosen distribution using a generative adversarial (GAN) training scheme [7, 25]. Conditional methods then provided the ability to change the style of an input image to another style [12, 20]. Initially such trained networks could only handle one style [43]; more recent works now allow multiple attribute changes using a single network, by learning to disentangle these attributes from the training images [16, 31, 44].

Novel view synthesis generates an image from a new, user-specified viewpoint, given one or more images of a scene from known viewpoints. We focus on methods that, like ours, can synthesize novel views from a single input image. This is a highly ill-posed problem, requiring strong 3D understanding and disentanglement of viewpoint and object shape from the input image. Since the seminal work of Hoiem *et al*. [11], methods have sought to develop more expressive models to address general NVS. Early CNN solutions regressed output pixel color in the new view [30, 41] directly from the input image. Some works disentangle their representations [31, 41], separating pose from object [41] or face identity [31]. Zhou *et al*. [42] introduced a flow prediction formulation, inferring an output to input pixel mapping instead, to which an explicit occlusion detection and inpainting module [23] and generalization to an arbitrary number of input images [29] have been added. Eslami *et al*. [3] developed a latent representation that can be *aggregated* to combine inputs, showing good results on synthetic geometric scenes.

A drawback of all these approaches is that they condition their networks to perform the transformation, limiting the transformations that can be applied to those that have been learned. Most recently, methods have been proposed to generate explicit representations of geometry and appearance that are transformed and rendered using standard rendering pipelines [19, 32]. While these representations can be rendered from arbitrary viewpoints, they are based on planar representations and are therefore not able to capture realistic shape, especially when rendered from side views. Our TBN approach allows us to perform fine-grained and varied, even non-rigid, 3D manipulations in the bottleneck volume, synthesizing them into realistic novel views. Here, the manipulations are applied manually. However, recent work [36] proposes a learned network for deforming objects arbitrarily (parameterized by an input shape), an idea that complements our framework.

2.2. Volumetric reconstruction and rendering

Several recent methods reconstruct an explicit occupancy volume from a single image [2, 5, 15, 27, 33, 39, 38, 40], some of which are trained using only supervision from 2D images [27, 33, 40]. Yan *et al*. [40] max-pool occupancy along image rays to produce segmentation masks, and minimize their difference w.r.t. the ground-truths. Tulsiani *et al*. [33] enforce photo-consistency between projected color images (given the camera poses) using the correspondences

[2] Image to depth map [6, 17], 3D mesh [10, 13, 35], point cloud [4] and surfel primitive [8] approaches also exist, but are outside the scope of our discussion.

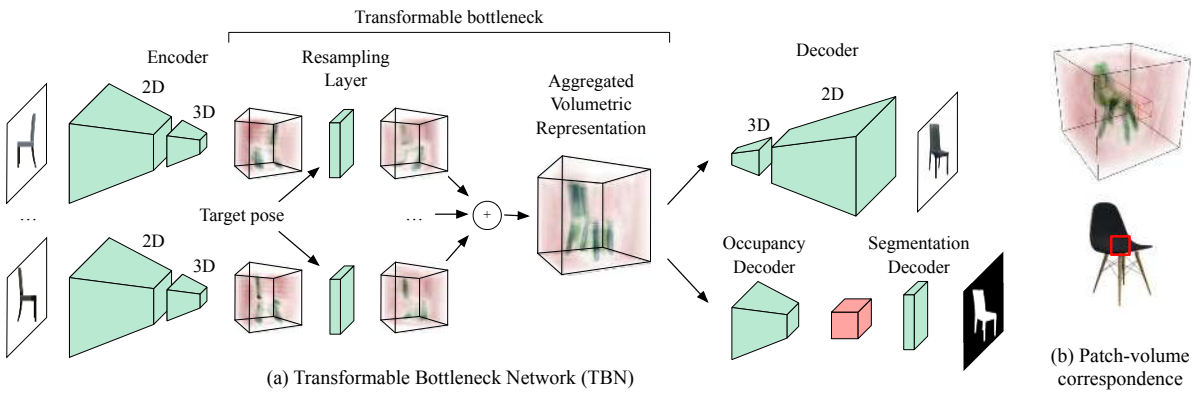


Figure 2: **A Transformable Bottleneck Network**. (a) Network architecture, consisting of three parts: an encoder (2D convolution layers, reshaping, 3D convolution layers), a resampling layer, and a decoder (a mirror of the encoder architecture). The encoder and decoder are connected purely via the bottleneck; no skip connections are used. The resampling layer transforms an encoded bottleneck to the target view via trilinear interpolation. It is parameterless, *i.e*. transformations are applied explicitly, rather than learned. Multiple inputs can be aggregated by averaging bottlenecks prior to decoding. (b) A visualization of the conceptual correspondence between an image patch and a subvolume of the bottleneck. Bottleneck volume visualizations show the cellwise norm of feature vectors. It is interesting that to note that this norm appears to encode the object shape.

implied by the occupancy volume. In contrast to these approaches that use explicit occupancy volumes and rendering techniques, the implicit approaches proposed by Kar *et al*. [15], and in particular Rezende *et al*. [27], are more relevant to our work—both the volumetric representation and the decoder (rendering) are learned, similar to recent neural rendering work [22]. The former [15], trained on ground truth geometry to estimate geometry from images[3], uses three learned networks[4] and a hand-designed unprojection step to compute a latent volume. The latter [27] requires the target transformation to be inferred by the network for NVS, whereas ours requires it to be provided as input, removing any limitations on the transformations that can be applied at test time.

3. Transformable bottleneck networks

In this section we formally define our Transformable Bottleneck Network architecture and training method.

3.1. Architecture

A TBN architecture (Fig. 2(a)) consists of three blocks:

1. An encoder network $\mathrm{E} : I_k \rightarrow \mathbf{X}_k$ with parameters θ_E, that takes in an image I_k and, through a series of 2D convolutions, reshaping, and 3D convolutions[5], outputs a bottleneck representation, $\mathbf{X}_k$, structured as a volumetric grid of cells, each containing an n-dimensional feature vector.
2. A *parameterless* bottleneck resampling layer $\mathrm{S} : \mathbf{X}_k, F_{k\rightarrow l} \rightarrow \mathbf{X}'_l$, that takes a bottleneck representation and user-provided transformation parameterization, $F_{k\rightarrow l}$, as input, and transforms the bottleneck via a trilinear resampling operation.
3. A decoder network $\mathrm{D}_I : \mathbf{X}'_l \rightarrow I'_l$ with parameters θ_I, whose architecture mirrors that of the encoder, that decodes the transformed bottleneck, $\mathbf{X}'_l$, into an output image, I'_l.

Subscripts k and l represent viewpoints. Neither the encoder nor the decoder are trained to perform a transformation: it is fully encapsulated in the bottleneck resampling layer. As this layer is parameterless, the network cannot *learn* how to apply a particular transformation at all; rather, it is applied explicitly. A single source image synthesis operation, which is end-to-end trainable, is written as:

$$I'_l = \mathrm{D}_I(\mathrm{S}(\mathrm{E}(I_k, \theta_E), F_{k\rightarrow l}), \theta_I). \tag{1}$$

When $F_{k\rightarrow l}$ is the identity transform (*i.e*. $k = l$), this operation defines an auto-encoder network.

3.1.1. Handling multiple input views

Our formulation naturally extends to an arbitrary number of inputs, both for training and testing, without modifications to either encoder or decoder. The encoded and transformed representations of all inputs are simply averaged:

$$\mathbf{X}'_l = \frac{1}{|K|} \sum_{k \in K} \mathrm{S}(\mathbf{X}_k, F_{k\rightarrow l}), \tag{2}$$

where K is the set of input viewpoints. The number of inputs tested on can differ from the number trained on, which can differ even within a training batch. We later show that the model trained with a single input view can effectively aggregate multiple inputs at inference time, and also that a model trained on multiple inputs can perform state-of-the-art inference from a single image.

[3]The latent representation therefore does not encode appearance.
[4]For 2D image encoding, recurrent fusion and a 3D grid reasoning.
[5]See the appendix for the exact architecture.

3.1.2. Bottleneck layout and resampling

The network architecture defines the number of cells along each side of the bottleneck volume, but not the spatial position of each cell. Indeed, the framework imposes no constraints on their position, *e.g*. the voxel grid cells do not need to be equally spaced. In this work the grid cells are chosen to be equally spaced[6], with the volume centered on the target object and axis aligned with the camera coordinate frame. Perspective effects caused by projection through a pinhole camera, and the camera parameters that affect them (such as focal length), are learned in the encoder and decoder networks, rather than handled explicitly.

Since the bottleneck representation is a volume, it can be resampled via trilinear interpolation, which is fully differentiable [14, Eqn. 9]. This allows it to be spatially transformed. The transformation, $F_{k\to l}$, is parameterized as a flow field that, for each output grid cell, defines the 3D point in the input volume to sample to generate it. The decoder takes as input a volume of the same dimensions as the encoder produces, therefore the flow field also has these dimensions. Feature channels form separate volumes that are resampled independently, then recombined to form the output volume.

When the view transformation is rigid, as in the case of NVS, the flow field is computed by transforming the cell coordinates of the novel view by the inverse of the relative transformation from the input view[7]. Non-rigid deformations can also be applied, enabling creative shape manipulation, which we demonstrate in Sec. 4.4. Importantly, we do not train on these kinds of transformations.

3.1.3. Geometry decoder

Since the TBN spatially disentangles shape and appearance within the volumetric bottleneck, it should also be able to reconstruct an object in 3D from the bottleneck representation. Indeed, prior work [27, 33] shows that training a 3D reconstruction using the NVS task alone, *i.e*. without 3D supervision, is possible. We extract shape in the form of a scalar occupancy volume, O, with one value per bottleneck cell, using a separate, shallow network, occupancy decoder, $\mathrm{D}_O : \mathbf{X} \to O$. To avoid using any 3D supervision to train this decoder, we then apply another decoding layer, $\mathrm{D}_S : O \to S$, that applies a 1D convolution along the z-axis (the optical axis), followed by a sigmoid, to generate a scalar segmentation image S, thus:

$$S = \mathrm{D}_S(O, \theta_S), \quad O = \mathrm{D}_O(\mathbf{X}, \theta_O), \tag{3}$$

where θ_O and θ_S are the parameters of the occupancy and segmentation decoders respectively.

3.2. Training

We train the TBN using the NVS task as follows.

[6]The scale of the spacing is unimportant here, as our NVS experiments only involve camera rotations around the object center.

[7]The flow is defined from output voxel to input voxel coordinate.

3.2.1. Appearance supervision

NVS requires a minimum of two images of a given object from different, known viewpoints[8]. Given $\{I_k, I_l\}$ and $F_{k\to l}$, we can compute a reconstruction, I'_l, of I_l using equation (1). Using this, we define several losses in image space with which to train our network parameters. The first two are a pixel-wise L_1 reconstruction loss and an L_2 loss in the feature space of the VGG-19 network, often referred to as the perceptual loss:

$$\mathcal{L}_{\mathrm{R}}(\theta_E, \theta_I) = ||I_{k\to l} - I_l||_1, \tag{4}$$

$$\mathcal{L}_{\mathrm{P}}(\theta_E, \theta_I) = \sum_i ||\mathrm{V}_i(I_{k\to l}) - \mathrm{V}_i(I_l)||_2^2, \tag{5}$$

where V_i is the output of the i^{th} layer of the VGG-19 network. To enforce structural similarity of the outputs we also adopt the structural similarity loss [28, 37], denoted as $\mathcal{L}_{\mathrm{S}}$. Finally, we employ the adversarial loss of Tulyakov *et al*. [34], $\mathcal{L}_{\mathrm{A}}$, to increase the sharpness of the output image.

3.2.2. Segmentation supervision

Appearance supervision is sufficient for NVS tasks, but to compute a 3D reconstruction we also require segmentation supervision[9], in order to learn θ_O and θ_S. We therefore assume that for each image I_i we also have a binary mask M_i, with ones on the foreground object pixels and zeros elsewhere![10] Segmentation losses are computed in *all* input and output views, using the aggregated bottleneck in the multi-input case, as follows:

$$\mathcal{L}_{\mathrm{M}}(\theta_E, \theta_O, \theta_S) = \sum_{k\in K} H(\mathrm{D}_S(\mathrm{S}(O_l, F_{l\to k}), \theta_S), M_k), \\ + H(\mathrm{D}_S(O_l, \theta_S), M_l), \tag{6}$$

where $O_l = \mathrm{D}_O(\mathbf{X}'_l, \theta_O)$ and H is the binary cross entropy cost, summed over all pixels. Summing over all views achieves a kind of space carving. Correctly reconstructing unoccupied cells within the visual hull is difficult to learn as no 3D supervision is used, but appearance supervision helps address this.

3.2.3. Optimization

The total training loss, with hyper-parameters λ_i to control the contribution of each component, is

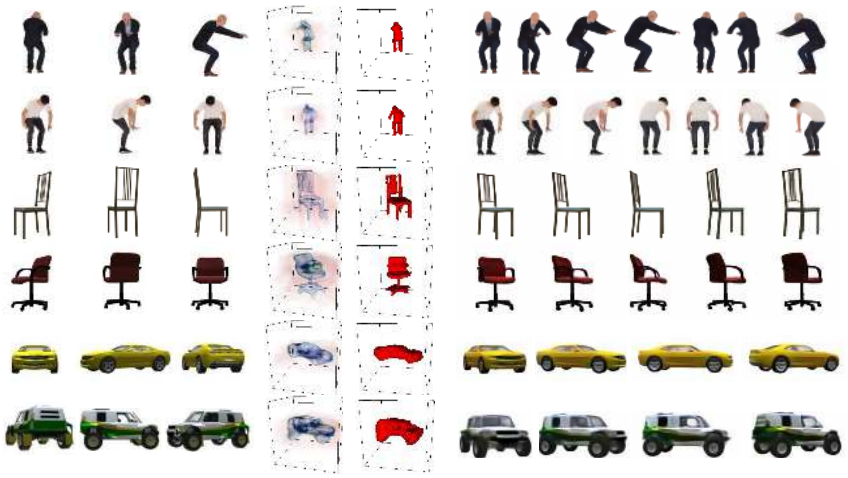
$$\mathcal{L}_{\mathrm{T}}(\Theta) = \mathcal{L}_{\mathrm{R}} + \lambda_1\mathcal{L}_{\mathrm{P}} + \lambda_2\mathcal{L}_{\mathrm{S}} + \lambda_3\mathcal{L}_{\mathrm{A}} + \lambda_4\mathcal{L}_{\mathrm{M}}, \tag{7}$$

This loss is fully differentiable, and the network can be trained end-to-end by minimizing the loss w.r.t. the network parameters $\Theta = \{\theta_E, \theta_I, \theta_O, \theta_S\}$ using gradient descent.

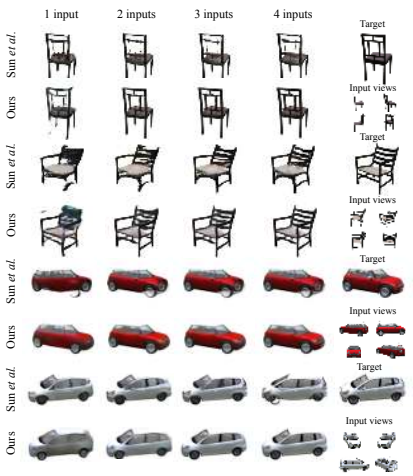
[8]Viewpoints are defined by camera rotation and translation, w.r.t. some arbitrary reference frame; world coordinates are not required.

[9]3D supervision could be used, but requires ground truth 3D data.

[10]Segmentation supervision is not a hard constraint, therefore segmentations from state-of-the-art methods (*e.g*. Mask R-CNN [9]) may suffice. However, we use ground truth masks in this work.



(a) Results on novel view synthesis



(b) Comparisons with Sun *et al*. [29]

Figure 3: **Qualitative results and comparisons.** (a) Randomly selected NVS samples generated using our method. *Left*: input images (3 of the 4 used). *Middle*: transformable bottleneck and 3D reconstruction. *Right*: synthesized output views. (b) Samples of synthesized novel views using the method of Sun *et al*. [29] and ours. Their method fails to capture overall structure for chairs, and generates unnatural artifacts on cars, especially around the wheels. Where < 4 input views are used, they are selected in clockwise order, starting top left.

4. Experiments

We train and evaluate our framework on a variety of tasks. We provide quantitative evaluations for our results for novel view synthesis using both single and multi-view input, and compare our results to state-of-the-art methods on an established benchmark. We also perform 3D object reconstruction from a single image and quantitatively compare our results to recent work [33]. Finally, we provide qualitative examples of our approach applying creative manipulations via non-rigid deformations.

4.1. A note on implementation

Our models are implemented and trained using the PyTorch framework [24], for automatic differentiation and parallelized computation for training and inference. We extended this framework to include a layer to perform parallelizable trilinear resampling of a tensor, in order to efficiently perform our spatial transformations. We plan to release the source code for our framework to the research community upon publication.

Each network was trained on 4 NVIDIA P100s, with each batch distributed across the GPUs. As we found that batch size had no discernible effect on the final result, we selected it to maximize GPU utilization. We trained each model until convergence on the test image set, which took approximately 8 days. For more details on the network architecture, training process and datasets used in our evaluations and results, please consult the appendix.

4.2. Novel view synthesis

Setup. We use renderings of objects obtained from the ShapeNet [1] dataset, which provides textured CAD models from a variety of object categories. We measure the capability of our approach to synthesize new views of objects under large transformations, for which ground-truth results are available. We train and evaluate our approach using the cars and chairs categories, to demonstrate its performance on objects with different structural properties. Each model is rendered as 256×256 RGB images at 18 azimuth angles sampled at 20-degree intervals and 3 elevations (0, 10 and 20 degrees), for a total of 54 views per model. We use standard training and test data splits [23, 29, 42], and train a separate network for each object category (also standard), using 4 input images to synthesize the target view. The network architecture and training method were fixed across categories.

As described in Section 3.1.1, our framework can use a variable number of input images. Though trained with 4 input images, we demonstrate that our networks can infer high-quality target images using fewer input images at test time. Using the experimental protocol of Sun *et al*. 2018 [29], which uses up to 4 input images to infer a target image, we report quantitative results for our approach and others that can use multiple input images [29, 30, 42], as well as for an approach accepting single inputs [23].

To further demonstrate the applicability of our method to non-rigid objects with higher pose diversity and lower appearance diversity, we also train and qualitatively evaluate a network using a multi-view human action dataset [26]. This dataset uses a limited number (186) of textured CAD models representing human subjects. However, the subjects are rigged to perform animation sequences representing a variety of common activities (running, waving, jumping, *etc*.), resulting in a much larger number of renderings. Note

	Methods	Car		Chair	
		L_1	SSIM	L_1	SSIM
1 view	Tatarchenko *et al*. 2015 [30]	.139	.875	.223	.882
	Zhou *et al*. 2016 [42]	.148	.877	.229	.871
	Park *et al*. 2017 [23]	.119	.913	.202	.889
	Sun *et al*. 2018 [29]	.098	.923	.181	**.895**
	Ours	**.091**	**.927**	**.178**	**.895**
2 views	Tatarchenko *et al*. 2015 [30]	.124	.883	.209	.890
	Zhou *et al*. 2016 [42]	.107	.901	.207	.881
	Sun *et al*. 2018 [29]	.078	.935	.141	.911
	Ours	**.072**	**.939**	**.136**	**.928**
3 views	Tatarchenko *et al*. 2015 [30]	.116	.887	.197	.898
	Zhou *et al*. 2016 [42]	.089	.915	.188	.887
	Sun *et al*. 2018 [29]	.068	.941	.122	.919
	Ours	**.063**	**.943**	**.116**	**.936**
4 views	Tatarchenko *et al*. 2015 [30]	.112	.890	.192	.900
	Zhou *et al*. 2016 [42]	.081	.924	.165	.891
	Sun *et al*. 2018 [29]	.062	**.946**	.111	.925
	Ours	**.059**	**.946**	**.107**	**.939**

Table 1: **Quantitative results on novel view synthesis**. We report the L_1 loss (lower is better) and the structural similarity (SSIM) index (higher is better) for our method and several baseline methods, for 1 to 4 input views, on both car and chair ShapeNet categories.

that the training process is identical to that used for rigid objects—input images for a given scene see the subject in a *fixed* pose. Thus, the capability to perform non-rigid transformations, as seen in Sec. 4.4, is still implicitly learned by the network.

Results. Table 1 reports quantitative results across recent methods, for 1 to 4 input views, on car and chair categories, for both the L_1 cost and structural similarity (SSIM) scores [37]. Though our networks are trained using exactly 4 input views, we obtain state-of-the-art results across *all* metrics, categories and number of input views, even in the challenging case of single-view input.

These results indicate that the TBN excels at NVS, and outperforms alternatives using both pixelwise and perceptual metrics. We further note that our method performs significantly better than others in cases involving large transformations of the input images and challenging viewpoints (see Fig. 3b). This demonstrates that our approach to combining information from these viewpoints is an effective strategy for synthesizing novel viewpoints, in addition to having other interesting applications (see below).

Fig. 3a shows qualitative examples on 3 datasets: the ShapeNet cars and chairs used for our quantitative evaluations, and the aforementioned human activity dataset. Fig. 3b qualitatively compares our results with those of Sun *et al*. [29] on several challenging examples requiring large viewpoint transformations from the chair and car datasets. Their method has difficulty inferring the proper correspondence between the source and target images for both object categories, particularly the more complex and variable structure of the chairs. Thus, many details are missing or incorrectly transformed. For cars, errors in the correspondence between local regions of source and target images cause artifacts, such as the wheel on the front of the car in row 5. In contrast, our method recovers the overall structure of both chairs and cars well, improving finer details as additional input views are added. We note that their results are in some cases sharper, as they use flow prediction to directly sample input pixels to construct the output, whereas our output images are rendered entirely from the bottleneck representation, as is required for general 3D manipulation.

4.3. Appearance synthesis for 3D reconstruction

As reported above, our method performs well on NVS with a single view, and progressively improves as more input views are used. We now show that this trend extends to 3D reconstruction. However, given that more views aid reconstruction, and that our network can generate more views, an interesting question is whether the generative power of our network can be used to aid the 3D reconstruction task. We ran experiments to find out.

Setup. To evaluate our method, we use the 3D reconstruction evaluation framework from the Differentiable Ray Consistency (DRC) work of Tulsiani *et al*. [33], which infers a 3D occupancy volume from a single RGB image. We trained our network on their dataset: multi-view images of ShapeNet objects, rendered under varying lighting conditions from 10 viewpoints, randomly sampled from uniform azimuth and elevation distributions with ranges $[0, 360)$ and $[-20, 30]$, respectively. As our method is trained using a set of multi-view images and corresponding segmentation masks, we compare our method to their publicly available model trained on masked, color images, using 5 random views of each object. In contrast, for this task our model was trained using only 2 random views (one input, one output) of each object.

Using the DRC [33] experimental protocol, we report the mean intersection-over-union (IoU) of the volumes from our occupancy decoder, computed on the evaluation image set, compared to the ground-truth occupancies obtained by voxelizing the 3D meshes used to render these images. Like DRC, we report the IoU attained using the optimal discretization threshold for each object category.

Results. Figure 4 shows the results of this evaluation. We report IoU numbers obtained using one real input image, with 0 to 9 additional synthesized views, sampled either randomly (red line) or regularly (at 0° elevation, blue line). For comparison, we show results using additional real images of the target object (green line), randomly sampled from the evaluation set (regularly sampled images were not available), as well as the results using DRC [33] with a single input image (yellow line). The figure also contains qualitative comparisons of results[11] using our best method (regularly sampled synthetic images) with varying numbers of synthetic images (middle columns), compared to DRC [33] (left) and the ground truth (right). Our method produces

[11] We render the voxel grids as meshes using an isosurface method.

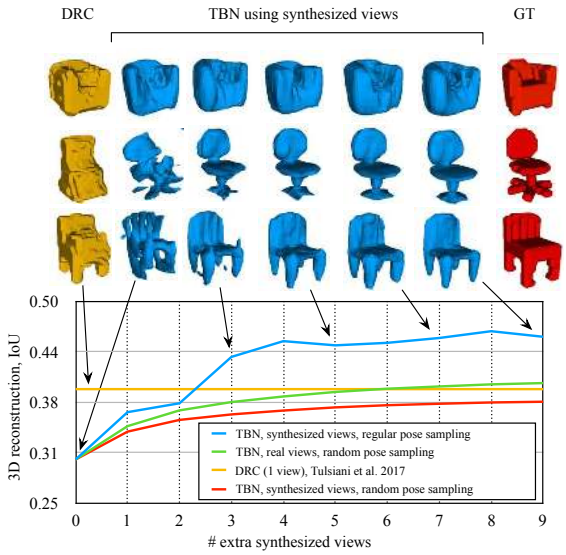


Figure 4: **3D reconstruction results**. Quantitative (IoU, following the evaluation framework of Tulsiani *et al*. [33]) and qualitative results of our method performing 3D reconstruction on the chairs dataset, from a single input image, supplemented by additional views synthesized by our network. 0 synthesized views indicates that only the original input image is used, while 1 to 9 indicate that we synthesize these additional views and combine the bottlenecks generated from these viewpoints with those obtained from the original input view. Results from Tulsiani *et al*. [33], who use only one image during inference, are also shown.

good results even with concavities (Fig. 4, row 1), that could not be obtained solely from the object's silhouette, demonstrating that NVS supervision is an able substitute for geometry supervision when inferring the geometric structure of such objects.

Using synthesized views from random poses clearly improves the reconstruction quality as more views are incorporated into our representation, though does not match the quality attained when using the same number of real images instead. Using synthetic views sampled at *regular* intervals around the object's central axis produces significantly better results, achieving superior single view 3D reconstruction to all other methods when using as few as 3 synthetic views. This dramatic improvement from randomly to regularly sampled synthetic views can be explained by the fact that information from each of the regularly sampled views is much more complementary than for the random views, that could leave parts of the object "unseen" (or unhallucinated). That synthetic views should improve the results at all is a more nuanced argument.

One might imagine that recycling hallucinated views into the encoder would simply reinforce the existing reconstruction. However, we argue the following: the encoder learns to extract the features that allow an image to be transformed, and the decoder learns to process the transformed features so as to produce a plausible image under this transformation. Therefore, consider a chair viewed from only one angle: the encoder could say where in space it believes the *visible* parts be, allowing it to be transformed, then the decoder could see this partial reconstruction in the bottleneck, and knowing what chairs look like, hallucinate the *unseen* parts. By recycling the synthesized image back through the encoder, it could then see new parts of the chair, and generate structure for them also. In essence, it comes down to where unseen structure is hallucinated within the network. Since the bandwidths of our encoder and image decoder are identical, there is no reason for it be in any particular part. However, because the gradients in the decoder layers have been passed through fewer other layers, they may receive a stronger signal for hallucination from the output view, hence learn it first.

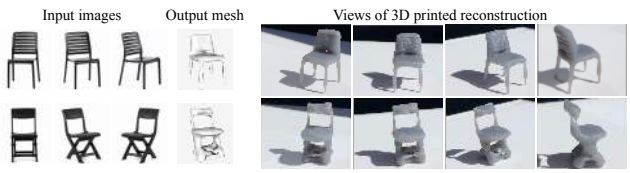


Figure 5: **Examples of 3D printed objects** created using our approach to 3D reconstruction.

One might expect the occupancy decoder to learn to hallucinate structure as well as the image decoder, but our results indicate that it doesn't (see our qualitative reconstructions with no synthetic views, in Fig. 4). We intuit that this is because it has much less information (binary *vs*. color images) to train on, and concomitantly a significantly smaller bandwidth. This further validates our hypothesis that appearance supervision improves 3D reconstruction within the visual hull, in the absence of 3D supervision.

Physical recreations of real objects. An exciting possibility of image-based reconstruction is being able to recreate old objects from photographs. We took 3 photos each of 2 *real* chairs, computed TBNs from these images and aggregated them using estimated relative poses. We computed occupancy volumes from these, extracted meshes using an isosurface method, and 3D printed these meshes. Figure 5 shows the input images, reconstructed meshes and 3D printed objects. Despite the low resolution of the occupancy volume (40^3 voxels), these physical recreations are coherent and depict the salient details of each chair.

4.4. Non-rigid transformations

Spatial disentanglement. Due to the convolutional nature of our network, a subvolume of the 3D bottleneck broadly corresponds to a patch of the input (if encoding) or output (if decoding) image, as visualized in Fig. 2(b). Any of the features in the subvolume, or a combination of them, can account for the appearance of the image patch; there is no guarantee that the features used will come from the voxels corresponding to the location in 3D space of the

Figure 6: **Creative, non-rigid manipulations** Selected examples of non-rigid 3D manipulations applied to transformable bottlenecks, for creative image and video[12] synthesis. Manipulations include: vertical and horizontal stretching, twisting, slicing & stitching, non-linear inflation. Please see the supplementary video for animated examples.

surface seen in the patch. In our framework, however, 2D supervision from multiple directions (both input and output views) places multiple subvolume constraints on where information can be stored. Storing information in the cells corresponding to the location in 3D space of the visible surface is the most efficient layout of information that meets all of those constraints, thus the one which achieves the lowest loss given the available network bandwidth. The effect is therefore achieved implicitly, rather than explicitly.

Creative manipulation. Based on this effect of spatial disentanglement, *arbitrary* non-rigid volumetric deformations can be applied on the transformable bottleneck, resulting in a similar transformation of shape of the rendered object. We demonstrate this qualitatively with a variety creative tasks, shown in Figure 6, that are performed by manipulating and combining the volumetric bottlenecks extracted from input images. Objects can be stretched in different dimensions (*first and second rows*). By rotating the upper and lower portion of the volume in opposite directions (*third row*), we can transform different regions of the target into a new shape that does not correspond to a single rigid transformation. Non-uniform and/or local scaling can be applied to inflate or shrink (*bottom row*) objects. Parts of a bottleneck can even be replaced with another part from the same, or a different bottleneck, creating hybrid objects (*fourth and fifth rows*). Many other such manipulations are possible, far beyond the scope of the rigid transformations trained on.

[12]While some such manipulations could *seem* simple to achieve in 2D, an edited 3D object can also be rendered consistently from any azimuth (see videos here and in the supplementary video), from a *single* manipulated bottleneck.

Figure 7: **Interactive manipulation**. We use our approach to rotate and deform objects before compositing them into real images.

Interactive creative manipulation. We implemented a tool to demonstrate a useful real-world application of the TBN: interactive manipulation and compositing. The user has one or more[13] photos of an object (whose class has been trained on) they wish to manipulate and place in a photo of a real world scene. The images are loaded into our application, from which a single aggregated bottleneck is computed. An interactive interface then allows the user to rotate, translate, scale and stretch the object, transforming and rendering the bottleneck in realtime and overlaying the object in the target image, as they apply the transformations.

Figure 7 contains example inputs and outputs of this process, for an interior design visualization use case. Two photos of a real chair were provided (with estimated relative pose). Rotations and stretches were then applied interactively, to get a feel for how the chair would look with different orientations and styles. Despite the challenging nature of this example (real photos of a chair with complex structure, and real-world lighting conditions such as specular highlights), we achieve highly plausible results.

5. Conclusion

This work has presented a novel approach to applying spatial transformations in CNNs: applying them directly to a volumetric bottleneck, within an encoder-bottleneck-decoder network that we call the Transformable Bottleneck Network. Our results indicate that TBNs are a powerful and versatile method for learning and representing the 3D structure within an image. Using this representation, one can intuitively perform meaningful spatial transformations to the extracted bottleneck, enabling a variety of tasks.

We demonstrate state-of-the-art results on NVS of objects, producing high quality reconstructions by simply applying a rigid transformation to the bottleneck corresponding to the desired view. We also demonstrate that the 3D structure learned by the network when trained on the NVS task can be straightfowardly extracted from the bottleneck, even without 3D supervision, and furthermore, that the powerful generative capabilities of the complete encoder-decoder network can be used to substantially improve the quality of the 3D reconstructions by re-encoding regularly spaced, synthetic novel views. Finally, and perhaps most intriguingly, we demonstrate that a network trained on purely rigid transformations can be used to apply arbitrary, non-rigid, 3D spatial transformations to content in images.

[13]Multiple images require true or estimated relative poses.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] 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. In *Proceedings of the European Conference on Computer Vision*, 2016.

[3] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 605–613, 2017.

[5] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proceedings of the European Conference on Computer Vision*, 2016.

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the Neural Information Processing Systems Conference*, pages 2672–2680, 2014.

[8] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[10] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *Proceedings of the British Machine Vision Conference*, 2018.

[11] Derek Hoiem, Alexei A Efros, and Martial Hebert. Automatic photo pop-up. In *ACM Transactions on Graphics*, volume 24, pages 577–584. ACM, 2005.

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[13] Dominic Jack, Jhony K. Pontes, Sridha Sridharan, Clinton Fookes, Shirazi Sareh, Frederic Maire, and Anders Eriksson. Learning free-form deformations for 3d object reconstruction. *Proceedings of the Asian Conference on Computer Vision*, 2018.

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Proceedings of the Neural Information Processing Systems Conference*, 2015.

[15] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Proceedings of the Neural Information Processing Systems Conference*, pages 365–376, 2017.

[16] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *Proceedings of the Neural Information Processing Systems Conference*, pages 5967–5976, 2017.

[17] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018.

[18] Chen-Hsuan Lin and Simon Lucey. Inverse compositional spatial transformer networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[19] Miaomiao Liu, Xuming He, and Mathieu Salzmann. Geometry-aware deep network for single-image novel view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[20] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proceedings of the Neural Information Processing Systems Conference*, pages 700–708, 2017.

[21] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *Proceedings of the Neural Information Processing Systems Conference*, pages 405–415, 2017.

[22] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Proceedings of the Neural Information Processing Systems Conference*, 2018.

[23] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

[26] Renderpeople, 2018. `http://renderpeople.com`.

[27] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[28] Jake Snell, Karl Ridgeway, Renjie Liao, Brett D Roads, Michael C Mozer, and Richard S Zemel. Learning to generate images with perceptual similarity metrics. In *Proceedings of the IEEE International Conference on Image Processing*, 2017.

[29] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J. Lim. Multi-view to novel view: Syn-

thesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision*, 2018.

[30] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR abs/1511.06702*, 1(2):2, 2015.

[31] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *Proceedings of the European Conference on Computer Vision*, 2018.

[33] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[34] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[35] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision*, pages 52–67, 2018.

[36] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[37] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[38] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5d sketches. In *Proceedings of the Neural Information Processing Systems Conference*, 2017.

[39] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[40] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[41] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Proceedings of the Neural Information Processing Systems Conference*, 2015.

[42] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proceedings of the European Conference on Computer Vision*, 2016.

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[44] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: image generation with disentangled 3d representations. In *Proceedings of the Neural Information Processing Systems Conference*, pages 118–129, 2018.