DeepVoxels++: Enhancing the Fidelity of Novel View Synthesis from 3D Voxel Embeddings

T. He[1], J. Collomosse[2,3], H. Jin[2] and S. Soatto[1]

[1] University of California, Los Angeles.
[2] Creative Intelligence Lab, Adobe Research.
[3] CVSSP, University of Surrey, UK.
{simpleig,soatto}@cs.ucla.edu, {collomos,hljin}@adobe.com

Abstract. We present a novel view synthesis method based upon latent voxel embeddings of an object, which encode both shape and appearance information and are learned without explicit 3D occupancy supervision. Our method uses an encoder-decoder architecture to learn such deep volumetric representations from a set of images taken at multiple viewpoints. Compared with DeepVoxels, our DeepVoxels++ applies a series of enhancements: a) a patch-based image feature extraction and neural rendering scheme that halves the 2D U-Net parameter numbers, and enables neural rendering at high resolution; b) learned view-dependent feature transformation kernels to explicitly model perspective transformations induced by viewpoint changes; c) a recurrent-concurrent aggregation technique to alleviate single-view update bias of the voxel embeddings recurrent learning process. Combined with d) a simple yet effective implementation trick of frustum representation sufficient sampling, we achieve significantly improved visual quality over the prior deep voxel-based methods (33% SSIM error reduction and 22% PSNR improvement) on 360° novel-view synthesis benchmarks of diffuse objects.

1 Introduction

A physical scene is far more complex than any number of images of it, so one cannot just "reconstruct *it*." The question of how to evaluate a *model* of a scene depends on whether one has access to additional sensors (*e.g.* tactile), or prior knowledge (*e.g.* scale of objects). In the absence of any side information, the most fundamental measure of quality of a model built from data is its ability to predict data that the model can generate [1, 2]. Hence, view synthesis can be thought of as a fundamental step in building and evaluating models of physical scenes from images. There are also practical ramifications of novel-view synthesis to video compression, graphics rendering and reinforcement learning. Before deep learning, novel-view synthesis was either approached as a pipeline of motion estimation, sparse reconstruction, topology estimation, meshing, and texture mapping [3], or directly by resampling the plenoptic function [4]. More recently, Sitzmann *et al.* [5] proposed DeepVoxels – an approach employing a 3D grid of persistent features integrated over input images along with 2D lifting and projection networks.

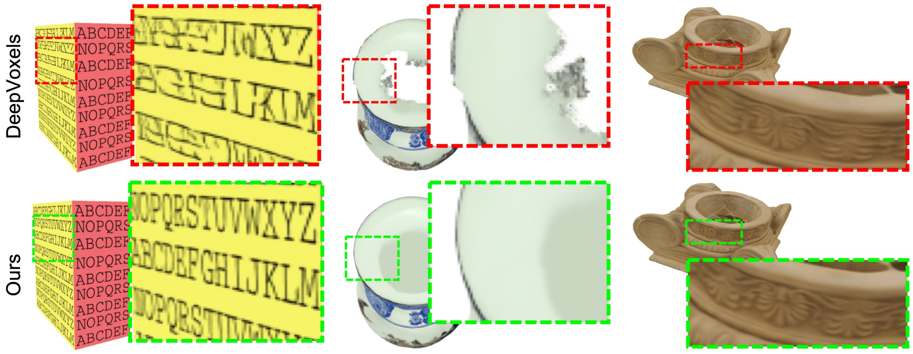

Fig. 1. Rendering results of our model DeepVoxels++ have sharper details (*e.g.* text, fine-grained shapes) and fewer artifacts (*e.g.* aliasing, holes) than DeepVoxels [5].

We learn 3D voxel embeddings of object shape and appearance based on image patches, whose pose can be directly controlled to generate novel views at high resolution. Our method is based on DeepVoxels [5], but with significantly improved rendering quality leveraging a series of improvements and a simple yet effective implementation trick of 3D embeddings sampling. Specifically, DeepVoxels++ is different from DeepVoxels in four aspects:

1. Low-complexity patch modelling. We adopt a patch-based training and inference scheme that halves the 2D U-Net parameters used in image feature extraction and neural rendering. It also reduces the complexity of large image context modeling (*e.g.* 512×512×3 full image vs. 128×128×3 small patch) and thus enables image modeling as well as rendering at high resolution in sliding window manner.

2. View-dependent voxel feature transformations. Viewpoint changes can cause perspective transformations in the observed images (see Fig. 3). We directly learn view-dependent feature transformation kernels in the lifting/projection steps to model such perspective effect. We transform the features from input patches to the 3D voxel embeddings and then from the voxels to output patches based on the relative voxel-camera poses. We demonstrate this idea on objects of diffuse reflection, delicate shapes and limited training views.

3. Recurrent-concurrent voxel feature aggregation. We aggregate 3D voxel embeddings utilizing both recurrent gated-fusion and concurrent max-pooling. It differs from existing works which treat multi-view images as a sequence and solely rely on recurrent networks [6, 5, 7]. Our method increases surface coverage of an object during each iteration of voxel feature aggregation and improves data utilization rate. For example, our model learned using 1/3 training data outperforms DeepVoxels [5] using the full data.

4. Frustum representation sufficient sampling. Sampling the 3D voxel embeddings into a frustum of the target pose is a critical step in decoding the learned volumetric representation into a rendered image. We empirically found that sufficient frustum sampling is a simple yet effective implementation trick to alleviate the issue of limited voxel resolution, reduce blurring artifacts, and preserve sharp details. It enforces the voxel feature learning process which in turn helps encode fine-scale details in the learned 3D voxel embeddings.

Overall, our approach DeepVoxels++ improves over DeepVoxels upon the visual quality of novel-view rendering at various poses (by up to 33% SSIM error reduction and 22% PSNR performance boost). We use the same 360° novel-view synthesis benchmarks as DeepVoxels [5], which contain 512×512 color images of delicate shapes/textures and diffuse reflectance. In contrast, other object based novel-view synthesis methods [8–11] mainly use the 256×256 ShapeNet images that consist of mostly mono-color flat surfaces and do not evaluate novel-view rendering results at 360° densely sampled poses. Finally, to add interpretability of our model, we conduct ablation studies to reveal the impact of our several enhancements in delivering this enhanced rendering quality.

2 Related work

Our work is related to multiple fields in 3D computer vision and graphics: image-based modeling, deep learning for view generation, 3D representation learning and neural-rendering, *and* deep learning with feature structure constraints. We mainly review papers in these areas.

Image-based modeling Image-based modeling and rendering techniques [3] are the early approaches to the novel view synthesis problem. Modern approaches, such as [12–14], are able to obtain high-quality results even for challenging scenarios with hand-held cameras. However these methods usually require multiple steps to (soft) reconstruct the object or learn image blending weights, and therefore they are prone to accumulative errors. They do not take full advantage of large scale multi-view datasets for 3D latent embedding learning and (adversarial) image generation training from the learned embeddings.

Deep learning for view generation With the advent of deep convolutional neural networks (CNNs), data-driven methods are gaining popularity for novel view generation [15, 16, 8, 17, 18, 9, 10, 19, 20, 5, 21–24]. The early methods overlook inherent 3D object structures/constraints and rely heavily on optical flow estimation [8] and generative adversarial networks [17, 18]. The former can maintain fine details in generated images while the latter are good at handling large pose changes for view synthesis. There are also hybrid approaches that combine the benefits of both sides [9, 10]. A common limitation of these methods is that they lack a geometrically persistent 3D representation of the object and thus tend to produce inconsistent images across output poses [5, 25].

3D representation learning and rendering 3D representation learning and neural-rendering with deep networks is a problem studied in 3D Deep Learning. Various approaches have been proposed using point clouds [26], implicit neural functions [21], voxel grids [27], multi-plane images [19, 20, 23, 28], and *etc.* We follow the line of work using voxel grids [27, 6, 11, 5] which offer a geometrically persistent structure to integrate visual information across multiple poses around the object. In particular Sitzmann *et al.* [5] demonstrate

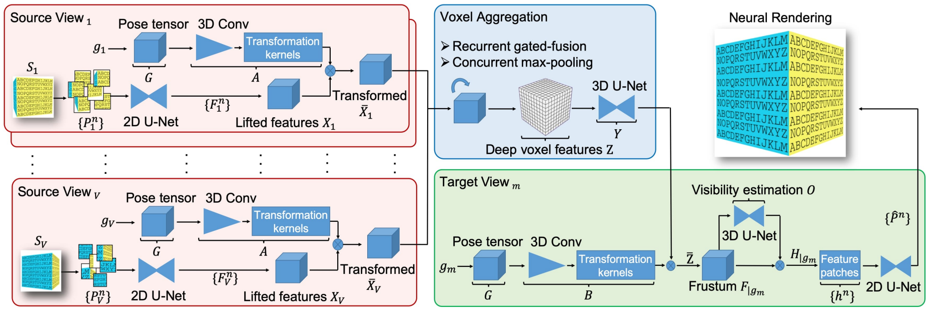

Fig. 2. DeepVoxels++ pipeline. Red: view-dependent patch feature extraction from V views. Blue: 3D voxel embeddings aggregation with recurrent gated-fusion and concurrent max-pooling. Green: view-dependent image patch rendering. Full network architectures are in the supplementary material. The networks are trained jointly with L^1 image reconstruction losses upon rendered views.

promising results for novel-view rendering utilizing a learned deep voxel representation. In this work, we achieve significantly enhanced visual quality for 360° novel-view synthesis than [5] via a series of enhancements on feature extraction/transformation/aggregation, and a simple yet effective implementation trick of voxel embeddings sufficient sampling when rendering images.

Learning with feature structure constraints Our work is also related to the emerging direction of introducing explicit structure constraints upon deep features to data-driven deep network models [29–32]. For example, Worrall *et al.* [30] impose a 3×3 rotation matrix constraint on high-dimensional features by length dividing and sub-vector multiplication to learn an interpretable representation for rotation/scaling factors. In this work, we propose to learn voxel feature transformation kernels conditioning on the relative voxel-camera poses. The learned kernels are used to model perspective transformations of the observed/rendered images induced by viewpoint changes under diffuse reflectance.

3 Method

Our model, DeepVoxels++, learns latent 3D voxel embeddings using color images of an object from multiple viewpoints. Our deep network architecture can be perceived as: an encoder-decoder with a geometrically consistent voxel feature space as the latent representation. As shown in Fig. 2, the architecture comprises three stages that are trained jointly by 2D view prediction without any 3D occupancy supervision: (*encoder*) view-dependent feature extraction from image patches, (*bottleneck*) recurrent-concurrent aggregation of lifted features to form the latent 3D voxel embeddings, (*decoder*) view-dependent patch rendering. At *test time* we only need the learned 3D voxel embeddings (*bottleneck*) and the view-dependent patch neural-rendering network (*decoder*) for 360° novel-view synthesis. Namely, we do not require any reference input image. Because the shape and appearance information of the target object has already been encoded into the learned volumetric features.

The training data of each object consists of M multi-view images $\{I_i, g_i\}_{i=1}^{M}$, where $I_i : D \subset \mathbb{R}^2$ is a 512×512×3 color image captured at a pose $g_i \in \mathbb{R}^{4\times 4}$. The pose can be computed by structure-from-motion (SFM) [33, 34]. At training time, the multi-view images are sampled into tuples of $\left\{S_i, T_i^0, T_i^1\right\}_{i=1}^{M}$. During each training step, the networks are updated with L^1 image reconstruction losses upon the predicted target views $\{(\hat{T}_j^0, \hat{T}_j^1)\}_{j=1}^{V}$, accepting multiple source images $\{S_j\}_{j=1}^{V}$ as input. Our approach aggregates information from V (*e.g.* 1, 4, 8, and *etc.*) views concurrently during training, making use of 3D-GRU and max-pooling at the bottleneck stage. This training methodology is to ensure large coverage of the object surface within each recurrent-concurrent step of 3D voxel embeddings aggregation. A degenerate case is DeepVoxels, which only conducts recurrent aggregation (strictly $V = 1$); *i.e.* without concurrent consideration of views. The previous training strategy induces single-view observation caused latent voxel feature update bias, and thus has low data utilization efficiency. For example, DeepVoxels++ learned using 1/3 training data outperforms DeepVoxels using the full data.

3.1 View-dependent patch feature extraction

To learn deep 3D voxel embeddings from multi-view images, we first sample image patches from training images in sliding window manner. We then extract 2D feature maps from the set of image patches and accumulate these features in voxel space of pre-defined resolution, via view-dependent feature lifting.

Patch feature extraction We subdivide each source image S_i into small-size patches $\{P_i^n\}_{n=1}^{N}$ via a sliding window with overlaps. Whilst early neural rendering papers used the ShapeNet dataset of 256×256×3 images [11, 9, 10], we use the recent, higher resolution 512×512×3 DeepVoxels dataset [5] which potentially requires more GPU memory to model/synthesize. Patches are encoded via a 2D U-Net with skip connections for feature extraction: $P_i^n \mapsto F_i^n$. For very large images, if GPU memory sizes prohibit training on all N patches at one pass, we can sample a subset $\{P_i^n\}_{n=1}^{N^{'}}$. In our experiments, we randomly sample 80% patches, but note the possibility of sampling heuristically *e.g.* sampling patches containing high-frequency/fine-scale content more frequently [35]. Compared with the full-image based prior methods, the patch-based scheme enables DeepVoxels++ to learn (and render) images of high resolution.

View-dependent feature lifting We first run SFM and compute the point cloud centroid in order to define s×s×s cubic voxels for aggregating feature patch lifting obtained voxel-shape features $X_i \in \mathbb{R}^{c\times s\times s\times s}$. By aligning the centroid with the voxel center and determining a suitable voxel scale, we get a voxel space tightly enclosing the object point clouds. Next, we project each voxel onto the feature patches $\{F_i^n\}_{n=1}^{N^{'}}$ and conduct differentiable bi-linear feature sampling to get the lifted voxel features. The projection operation, approximated via a pinhole camera model, is also differentiable. Note that the intrinsic matrix $K \in \mathbb{R}^{3\times 3}$

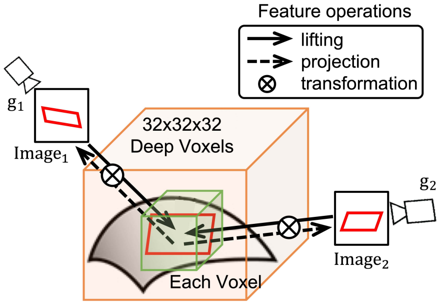

Fig. 3. A voxel that encodes a parallelogram pattern looks different at two poses due to perspective projection effects under diffuse reflectance.

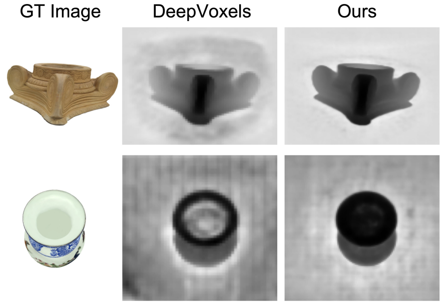

Fig. 4. Pseudo-depth maps visualized using estimated frustum visibility values. Our results are sharper than DeepVoxels and have less artifacts.

wrt. the image patches P_i^n has to be rectified to get K_r in order to correctly map world-coordinate locations onto *the extracted feature patches* F_i^n. Because an image patch and its corresponding feature patch have different sizes.

$$K_r = \begin{bmatrix} \alpha f_x & & \alpha c_x \\ & \beta f_y & \beta c_y \\ & & 1 \end{bmatrix} \tag{1}$$

where (f_x, f_y, c_x, c_y) belong to K. K_r is the rectified intrinsic matrix used in voxel projection, and (α, β) are (width, height) ratios between F_i^n and P_i^n.

The voxel space used to accumulate the lifted features X_i is typically of low resolution (*e.g.* 32×32×32). Therefore, each voxel can be considered to model a local surface region of the object, as illustrated in Fig. 3. It explains the motivation for perspective projection effect modeling by voxel feature transformations during the lifting (and projection) steps. We achieve this by applying learned convolutional feature transformation kernels $A(\cdot) \in \mathbb{R}^{c \times c \times 1 \times 1 \times 1}$ on the lifted features X_i.

$$\bar{X}_i = A(G(g_i)) \circledast X_i \tag{2}$$

where $\bar{X}_i \in \mathbb{R}^{c \times s \times s \times s}$ are the transformed features and $\circledast$ represents 3D convolution operation. As shown in Fig. 2, the kernel estimation network $A(\cdot)$ is implemented as several 3D convolution layers that take relative voxel-camera poses $G(g_i) \in \mathbb{R}^{6 \times s \times s \times s}$ as input and estimate convolutional feature transformation kernels. The reason why $G(g_i)$ is a 3D shape tensor is because each entry of it consists of the relative voxel-camera translation and the camera pose rotation vector. Note that voxels has different relative voxel-camera translations but share the same camera rotation vector. We adopt this encoding format of relative voxel-camera poses based on empirical studies.

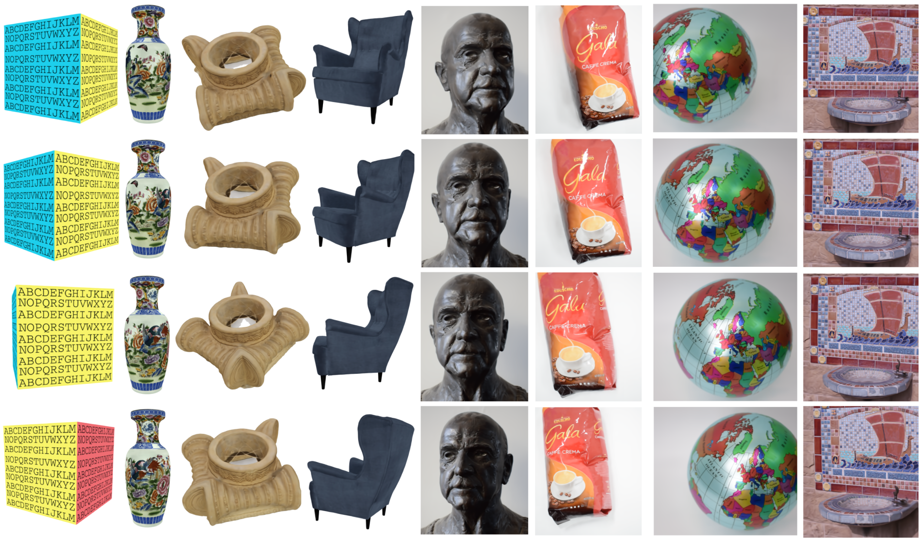

Fig. 5. Novel-view synthesis results of DeepVoxels++ on objects with large viewpoint changes and complex shape/texture patterns. Our model is proposed for diffuse objects but we also show a few preliminary results on objects with specularities and shadows.

3.2 Recurrent-concurrent voxel feature aggregation

The lifted and transformed features $\bar{X}_i$ from one source image S_i only provide a single-view observation of the object at pose g_i. To learn holistic 3D voxel embeddings $Z \in \mathbb{R}^{c\times s\times s\times s}$ we need to integrate features extracted from all the training views, which have about 500 images and thus cannot be aggregated into the voxels at one time. We address this challenge by aggregating $\{\bar{X}_j^k\}_{j=1}^V$ from V different views within each iteration (indexed by k) of voxel representation updates, via both recurrent gated-fusion and concurrent max-pooling. Note that the prior methods only integrate features from a single-view into Z at each feature update iteration, and therefore suffer from single-view observation bias. Our aggregation approach provides a large surface coverage of the object during each voxel representation update and improves data utilization rate. This enables our model, DeepVoxels++, to use less data and still achieve better rendering quality than DeepVoxels.

Recurrent gated-fusion We first use 3D-GRU [7] to separately fuse each single-view obtained $\bar{X}_j^k$ into the holistic 3D voxel embeddings Z^{k-1} that are obtained from the previous training iteration: $Z_j^k = \mathrm{GRU}(\bar{X}_j^k, Z^{k-1})$. Namely, the 3D object representation Z is modeled as the hidden voxel embeddings of 3D-GRU and will be recurrently updated when more views come in. At the first round of voxel representation aggregation, we initialize Z^0 with zeros. However, within each step of voxel embedding update, recurrent gated-fusion only aggregates features from a single-view observation. To tackle the problem of single-view update bias, we further utilize a multi-view based max-pooling operation upon the 3D voxel embeddings.

Concurrent max-pooling Now we need to aggregate a set of deep voxel embeddings $\{Z_j^k\}_{j=1}^V$ obtained separately by recurrent gated-fusion from $V > 1$ views. Inspired by Multi-view CNN [36], we use the max-pooling operation: $Z^k = \text{Max}(Z_1^k, Z_2^k, ..., Z_V^k)$. $\text{Max}(\cdot)$ means applying max-pooling operations along the first dimension (*i.e.* the feature channel) of the 3D voxel embeddings $Z_j^k \in \mathbb{R}^{c\times s\times s\times s}$. The obtained latent voxel representation Z^k (*i.e.* the 3D-GRU hidden embedding at the k-th iteration) will be passed into the next iteration of recurrent-concurrent voxel feature update until the end of training.

3.3 View-dependent patch rendering

Rendering a target image $T_{|g_m}$ from the 3D voxel embeddings $Z_{(j)}^{(k)}$[4] at any given pose g_m around the object involves three steps: view-dependent frustum feature sampling, depth dimension reduction and patch-based neural rendering.

View-dependent frustum sampling For each target camera pose g_m, we define a $d\times h\times w$ frustum space to enclose the $s\times s\times s$ cubic voxels where the volumetric embeddings Z are saved. We emphasize that while voxels are usually of low spatial resolution (*e.g.* 32×32×32) due to GPU memory size constraint, the rendering visual quality from these deep voxel embeddings can be substantially improved by sufficient frustum sampling. Namely, we utilize large 2D sampling sizes $h\times w$ (*e.g.* 128×128 vs. 32×32). The depth axis d is collapsed when rendering image patches. We found that this is a simple yet effective implementation trick for deep voxels-based high quality view synthesis. Ablation studies in the experiments section support this argument (see Fig. 7 and Tab. 3). Specifically, we can map the frustum into the voxel space by inverse-perspective projection and sample the transformed voxel features $\bar{Z} \in \mathbb{R}^{c\times s\times s\times s}$ by differentiable trilinear interpolation.

$$\bar{Z} = B(G(g_m)) \circledast Y(Z) \tag{3}$$

where $Y(\cdot)$ is a 3D U-Net that further refines the 3D voxel embeddings Z. As shown in Fig. 3, we need to conduct voxel feature transformations at both lifting and projection steps due to the corresponding perspective projection effect in the observed/rendered images. Thus, similar to Eq. 2, we use a kernel estimation network $B(\cdot)$ to directly take the relative voxel-camera poses $G(g_m) \in \mathbb{R}^{6\times s\times s\times s}$ as input and estimate convolutional feature transformation kernels. $B(\cdot)$ is also implemented as several 3D convolution layers. The sufficiently sampled frustum features from $\bar{Z}$ are denoted as $F_{|g_m} \in \mathbb{R}^{c\times d\times h\times w}$. Note that, as per Eq. 1, we use a rectified camera intrinsic matrix when conducting inverse-perspective projection for frustum representation sufficient sampling. In this case, scaling factors (α, β) are (width, height) ratios between the frustum and the target image.

[4] During training gradients are back-propagated to Z_j^k. At test time we use the converged Z for rendering. We use Z for convenience from here.

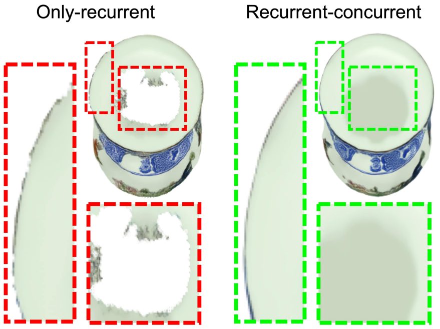

Fig. 6. 3D voxel embeddings aggregation: only-recurrent vs. recurrent-concurrent.

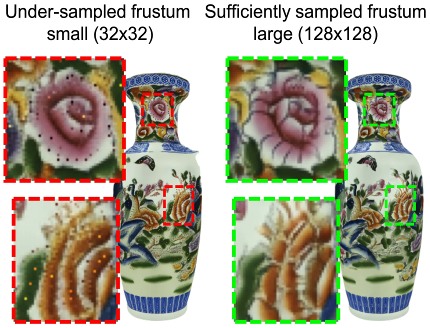
Under-sampled frustum small (32x32)
Sufficiently sampled frustum large (128x128)

Fig. 7. Frustum representation sampling sizes: small (32×32) vs. large (128×128).

Depth dimension reduction Rather than directly utilizing the frustum representation $F_{|g_m}$ for patch neural-rendering, we follow [5] and first collapse it into depth dimension reduced features $H_{|g_m} \in \mathbb{R}^{c\times h\times w}$ by weighted average feature pooling upon the depth dimension: $H_{|g_m} = \text{Avg}[F_{|g_m} \otimes O(F_{|g_m})]_{|dim=1}$. Here $\text{Avg}[\cdot]_{|dim=1}$ indicates weighted average feature pooling along the second dimension (*i.e.* depth) of the $c\times d\times h\times w$ input tensor. $\otimes$ means element-wise multiplication with broadcasting between $F_{|g_m} \in \mathbb{R}^{c\times d\times h\times w}$ and $O(\cdot) \in \mathbb{R}^{1\times d\times h\times w}$. $O(\cdot)$ is implemented as a 3D U-Net with skip connections, whose output can be treated as frustum visibility estimation *wrt.* a viewpoint g_m and adds interpretability to the rendering process. Because it enables the computation of pseudo-depth maps which explain several rendering artifacts of the prior methods (see Fig. 4). Specifically, inaccurate visibility estimation, induced by incorrectly up-weighting (in)visible surfaces and empty space within the frustum, can cause DeepVoxels' rendering artifacts like aliasing and holes.

Patch-based neural rendering The final step of the model is to render patches from $H_{|g_m}$. Recall that during the encoder stage, we explained the benefits of *patch-based feature extraction* (subsec. 3.1). During the decoding step we conduct patch-based neural rendering, for the same purposes of utilizing fewer 2D U-Net parameters, reducing the complexity of large image context modeling and being able to model/render images at high resolution. Similar to patch-based feature extraction, we subdivide $H_{|g_m}$ into small-size feature patches $\{h^n\}_{n=1}^{N}$ by a sliding window with overlaps, and then conduct patch neural rendering using a 2D U-Net with skip connections: $h^n \mapsto P^n$. At training time, we apply random sampling to retain $N^{'}$ feature patches (*e.g.* 80% of N) to save GPU memory. We use L^1 image reconstruction losses upon rendered image patches to enable joint training for the complete network architectures as shown in Fig. 2.

$$L(\hat{P}^n, P^n) = \frac{\sum_{n=1}^{N^{'}} \sum_{a,b} \left\| \hat{P}^n_{a,b} - P^n_{a,b} \right\|_1}{N^{'} * D} \tag{4}$$

Table 1. 360° novel-view synthesis benchmark of objects with diffuse reflectance. Higher values of PSNR and SSIM indicate better rendering quality. Our method DeepVoxels++ surpasses DeepVoxels and other competing methods by large margins.

	Vase	Pedestal	Chair	Cube	Mean
Method	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Nearest Neighbor	23.26 / 0.92	21.49 / 0.87	20.69 / 0.94	18.32 / 0.83	20.94 / 0.89
Tatarchenko *et al.* [39]	22.28 / 0.91	23.25 / 0.89	20.22 / 0.95	19.12 / 0.84	21.22 / 0.90
Worrall *et al.* [30]	23.41 / 0.92	22.70 / 0.89	19.52 / 0.94	19.23 / 0.85	21.22 / 0.90
Pix2Pix [40]	26.36 / 0.95	25.41 / 0.91	23.04 / 0.96	19.69 / 0.86	23.63 / 0.92
Neural Volumes [41]	20.39 / 0.84	36.47 / **0.99**	35.15 / **0.99**	26.48 / 0.96	29.62 / 0.95
DeepVoxels [5]	27.99 / 0.96	32.35 / 0.97	33.45 / **0.99**	28.42 / 0.97	30.55 / 0.97
Ours	**32.91** / **0.98**	**38.93** / 0.98	**40.87** / **0.99**	**36.51** / **0.99**	**37.31** / **0.99**

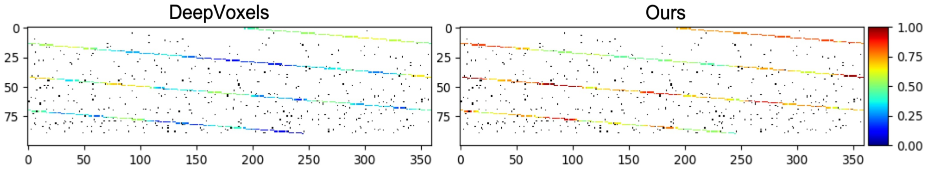

Fig. 8. Normalized azimuth-elevation PSNR maps on Cube. More objects are visualized in supplementary. Horizontal: $[0°, 360°]$ azimuth. Vertical: $[0°, 100°]$ elevation. Black dots are the training poses. Colored spiral lines are the test pose trajectories. Red color means large PSNR value and blue means small. These plots showcase smooth viewpoint interpolation paths between training views (*i.e.* black dots), showing consistent improvement over DeepVoxels across different novel views.

where $\hat{P}^n$ is a rendered image patch and P^n is a ground-truth patch. (a, b) are pixel indices within an image patch and D is the pixel number of a patch. At test time, we composit all N rendered patches $\{\hat{P}^n\}_{n=1}^N$ into the target image raster, and crop overlapped regions. The stitched patches comprise the final 512×512 color rendered image $\hat{T}_{|g_m}$.

3.4 Implementation details

We implement our approach using PyTorch [37]. The networks are trained with the ADAM optimizer [38] using an initial learning rate of 0.0004. For different benchmark objects, we use the same set of hyper-parameters and stop the training at 400 epochs, which takes about 4 days. However, to outperform the prior deep voxels-based methods we only need to train on *1/3 data* (as shown in Tab. 6) and therefore we are able to reduce the training time to around 1 day, while DeepVoxels' training takes about 3 days on *the full training dataset* and its results are much worse. More details of our network architectures can be found in supplementary.

Table 2. Better rendering quality can be achieved when more multi-view images V are aggregated in each round of recurrent-concurrent latent 3D voxel embedding updates.

V	Vase	Pedestal	Chair	Cube	Mean PSNR
1	27.99	32.35	33.45	28.42	30.55
4	**30.30**	34.64	35.97	**31.97**	33.22
8	29.45	**35.54**	**37.79**	31.65	**33.61**

4 Experiments and Discussion

We evaluate DeepVoxels++ on 360° novel-view synthesis benchmarks against several competing methods: a Nearest Neighbor baseline, Tatarchenko et al. [39], Worrall et al. [30], Pix2Pix [40], Neural Volumes [41], SRN [21] and DeepVoxels [5]. To add interpretability of our model, we also conduct ablation studies to reveal the impact of our series of enhancements in achieving this performance improvement.

4.1 Dataset and metrics

For fairness of comparison, we use the same dataset and evaluation metrics (*e.g.* the Structural Similarity Index (SSIM), the Peak Signal-to-noise Ratio (PSNR)) as DeepVoxels [5]. The dataset contains 512×512 color images of delicate shapes/appearance (*e.g.* pedestal, vase) and diffuse reflectance. For each object, the dataset has about 500 training images and 1000 test views as ground truth. The test views are densely sampled from a 360° spiral curve enclosing the object at different angles and distances, for evaluating smoothness and fidelity as the viewpoint changes. This contrasts with recent object based novel-view synthesis papers which mainly use the 256×256 ShapeNet image dataset. ShapeNet lacks the aforementioned appearance complexity, and does not evaluate novel-view rendering results at densely sampled test poses. Though not the purpose of our method, we also show a few results on objects with specular reflectance and shadows in order to shed light on future work.

4.2 Evaluating 360° novel view synthesis

Once trained on multi-view images of an object, our approach no longer requires those views at *test time* as reference-view input; a requirement of some recent methods [8–10]. Rather, we can directly use the learned 3D voxel embeddings Z that encode object shape and appearance to render high-resolution images from novel views. Tab. 1 shows our method DeepVoxels++ to outperform DeepVoxels [5] by 22% PSNR improvement and 33% SSIM error reduction. Both DeepVoxels and Neural Volumes [41] are based on (deep) voxel representations. Our method also surpasses a recent implicit neural representation method SRN [21] but it only reported mean results: 33.03 PSNR, 0.97 SSIM. We further visualize normalized azimuth-elevation PSNR maps in Fig. 8 to prove that our improvement is due to consistently improved rendering quality across 1000 dense test views of the object, not caused by over-fitting at certain viewpoints that are close to the training data. This capability to smoothly interpolate between training views at high fidelity contrasts with DeepVoxels.

Figs. 1 and 5 present rendering results on diverse objects. While the competing methods and our approach are proposed for and benchmarked on objects of diffuse reflectance, in the visualizations we also show some preliminary results on specular reflectance and shadow modeling. Our rendered images contain sharper details and fewer rendering artifacts such as blur, aliasing and holes than DeepVoxels, which we attribute to several features of our method. For example, Fig. 6 shows that recurrent-concurrent voxel aggregation can help reduce artifacts. Because it addresses single-view observation bias by increasing object surface coverage during each training iteration. Moreover, Fig. 7 indicates that frustum representation sufficient sampling is helpful in sharp details rendering because it alleviates the problem of limited voxel spatial resolution by enforcing strong supervision on 3D voxel embeddings (*i.e.* rich gradient signals). The computed pseudo-depth maps in Fig. 4 also explain some artifacts of the prior DeepVoxels method. Specifically, inaccurate visibility estimation, induced by incorrectly up-weighting (in)visible surfaces and empty space within the frustum, can cause view synthesis problems like aliasing and holes.

4.3 Ablation studies

Voxel feature aggregation Previous deep voxel methods [6, 5] use 3D-GRU [42, 43] for image-based modeling by adopting a structured voxel space as the hidden embedding and treating hundreds of multi-view images of an object as a video sequence. However, this type of single-view based *sequential* update manner can cause inefficiency and bias of 3D voxel embeddings learning. Because it imposes an ordering on viewpoints and biases training when only a single-view observation is aggregated during each recurrent step. Inspired by Multi-view CNN [36], we address these challenges by conducting voxel feature aggregation at two different dimensions jointly: recurrent gated-fusion and concurrent max-pooling. This provides a large surface coverage of the object during each iteration of voxel feature updates and improves data utilization rate. Results in Tab. 2 verify our arguments. Fig. 6 further demonstrates that our aggregation method helps to reduce DeepVoxels' rendering artifacts such as aliasing and holes. Better rendering quality and faster training can be achieved with more views aggregated by max-pooling in each round of voxel feature updates, which is most effective when view number increases from 1 to 4 and starts to become less effective when it reaches 8 views. Thus, in our benchmark results we use 4 views considering the trade-off between performance gains and GPU memory size constraints.

Frustum sufficient sampling To decode the learned 3D voxel embeddings and render an image at a target pose g_m, we need to first project the deep voxel features into a frustum. As explained in subsec. 3.3 the projection procedure essentially is feature sampling from the voxel space to the frustum space.

Table 3. Frustum representation sufficient sampling from the low spatial resolution deep voxels can substantially improve the 360° novel-view synthesis performance.

Sampling sizes $h\times w$	Vase	Pedestal	Chair	Cube	Mean PSNR
32×32	27.16	27.93	32.99	27.35	28.86
64×64	30.30	34.64	35.97	31.97	33.22
128×128	**32.62**	**38.75**	**38.73**	**35.35**	**36.36**

Table 4. Our patch-based scheme reduces the complexity of large image context modeling and improves the rendering results. It also halves the 2D U-Net parameters for image feature extraction and neural rendering.

		2D U-Net parameters (M)	
Method	Mean PSNR	Feature extraction	Neural rendering
Full-image	36.36	92.2	108.9
Patch	**36.99**	**40.3**	**56.9**

Table 5. Comparisons between without/with voxel feature transformations. With the learned feature transformation kernels, we achieve better performance on objects of delicate shapes (*e.g.* pedestal, chair) and limited training views (*e.g.* 30 images).

Method	Vase	Pedestal	Chair	Cube	Mean PSNR
Without	**26.05**	29.84	28.89	25.19	27.49
With	25.76	**30.83**	**29.45**	**25.43**	**27.87**

Tab. 3 and Fig. 7 show that while voxels are usually of $32\times32\times32$ low spatial resolution due to GPU memory constraints, sufficient frustum sampling can substantially improve the visual quality of rendered images with sharper details than DeepVoxels. The frustum representation sampling sizes are determined by height/width of the depth dimension reduced frustum feature maps. We use 128×128 sampling sizes in our main results. Frustum representation sufficient sampling is *a simple yet effective implementation trick* that addresses the low voxel resolution problem. One explanation is that though voxels have low spatial resolution, they contain high dimensional latent 3D embeddings, encoding objects' appearance and shape information. Meanwhile, the differentiable trilinear interpolation-based frustum sufficient sampling enforces strong supervision on the deep voxel features (*i.e.* rich gradient signals), and eventually helps to encode more fine-scale details into the learned latent 3D embeddings.

Low-complexity patch modeling The patch-based training/inference scheme has multiple advantages over the previous full-image based one, which are also demonstrated in other problems like point-cloud upsampling [44] and image restoration [45]. Besides reducing the complexity of modeling large image context and therefore improving fine-scale patch synthesis quality (as shown in Tab. 4), our approach requires only half the 2D U-Net parameters used in image feature extraction and neural rendering of prior methods due to small-size input. Furthermore, the patch-based scheme enables us to model and render images of high resolution at low GPU memory cost, whereas full-image based methods are not easily trainable with high resolution images.

Table 6. Our 1/3 data trained model can surpass full data trained DeepVoxels [5] by large gaps in mean PSNR. The results indicate that DeepVoxels++ is data efficient.

	Training data sizes			
Method	full	1/3	1/16	1/48
DeepVoxels	30.55	28.09	26.06	19.35
Ours	**37.31**	**33.34**	**27.87**	**20.71**

View-dependent voxel feature transformation Fig. 3 illustrates how 3D voxel embeddings that encode shape and appearance of an object's local surface plane can be mapped to different patterns, due to the corresponding perspective projection effect induced by viewpoint changes under diffuse reflectance. Such perspective transformation in the observed/rendered images is explicitly modeled by us leveraging learned feature transformation kernels from relative voxel-camera poses. In contrast, previous methods rely on voxel volume changes caused by vantage point changes to infer view-dependency. But voxel volume differences are constrained by low voxel spatial resolutions and only implicitly reflect viewpoints. Tab. 5 shows that explicit voxel feature transformation modeling is critical for objects with delicate shapes (*e.g.* pedestal) and limited training views (*e.g.* 30 images), where voxel volume changes are less continuous and less effective to model the corresponding perspective transformation caused by viewpoint changes under diffuse reflectance.

Number of training views While DeepVoxels requires around 500 multi-view images to learn faithful 3D voxel embeddings of an object, DeepVoxels++ can learn to synthesize high fidelity novel views even with a limited number of training images. In Tab. 6, we experiment on full-size, 1/3, 1/16 and 1/48 training data. Our method outperforms DeepVoxels in all conditions, demonstrating promising results for real-world applications where only few images are available for 3D object representation learning and novel view synthesis. For example, camera rig-based image capture systems. The improved data utilization rate is attributed to our recurrent-concurrent 3D voxel embedding aggregation method, as it alleviates single-view observation bias and therefore multi-view features can be extracted/accumulated into voxels efficiently.

5 Conclusion and Limitations

We have proposed a novel view modeling and rendering technique that learns latent 3D voxel embeddings from multi-view images of an object without 3D occupancy supervision. Our approach, DeepVoxels++, outperforms previous deep voxel-based methods by large margins on 360° novel-view synthesis benchmarks. We show that our novel view synthesis results contain more fine-scale details and less rendering artifacts than DeepVoxels [5]. We also conduct multiple ablation studies to show the impact of our series of improvements in achieving this enhanced rendering fidelity.

Although the benchmark mainly evaluates objects with diffuse reflectance, our proposed method of learning voxel feature transformation kernels potentially can also model other view-dependent effects (*e.g.* specularity) besides image-plane perspective transformations of diffuse surfaces. We demonstrate some preliminary visual results for specularity and shadow modeling in Fig. 5 and the supplementary video demo. But it is worth considering extending the current dataset with objects of non-Lambertian reflectance and conducting evaluations under various lighting situations. Novel-view rendering for non-rigid objects leveraging dynamic volumes is another challenging and important problem. In brief, future work could consider various scenarios that are not explicitly modeled or extensively evaluated by the current deep voxel-based methods, such as lighting and specular reflectance, multi-object scenes, dynamic objects and so on.

References

1. Anderson, B.D., Moore, J.B., Hawkes, R.: Model approximations via prediction error identification. Automatica **14** (1978) 615–622
2. Astrom, K.: Maximum likelihood and prediction error methods. IFAC Proceedings Volumes **12** (1979) 551–574
3. Kang, S.B., Li, Y., Tong, X., Shum, H.Y.: Image-based rendering. Foundations and Trends in Computer Graphics and Vision (2006)
4. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Siggraph. Volume 96. (1996) 43–54
5. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2437–2446
6. Tung, H.Y.F., Cheng, R., Fragkiadaki, K.: Learning spatial common sense with geometry-aware recurrent networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2595–2603
7. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
8. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: European conference on computer vision, Springer (2016) 286–301
9. Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: Transformation-grounded image generation network for novel 3d view synthesis. In: Proceedings of the ieee conference on computer vision and pattern recognition. (2017) 3500–3509
10. Sun, S.H., Huh, M., Liao, Y.H., Zhang, N., Lim, J.J.: Multi-view to novel view: Synthesizing novel views with self-learned confidence. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 155–171
11. Olszewski, K., Tulyakov, S., Woodford, O., Li, H., Luo, L.: Transformable bottleneck networks. arXiv preprint arXiv:1904.06458 (2019)
12. Hedman, P., Alsisan, S., Szeliski, R., Kopf, J.: Casual 3d photography. ACM Transactions on Graphics (2017)
13. Hedman, P., Philip, J., True Price, J.M.F., Drettakis, G., Brostow, G.J.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (2018)
14. Penner, E., Zhang, L.: Soft 3d reconstruction for view synthesis. In: ACM Transactions on Graphics. (2017)
15. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deepstereo: Learning to predict new views from the world's imagery. In: CVPR. (2016)

16. Ji, D., Kwon, J., McFarland, M., Savarese, S.: Deep view morphing. In: CVPR. (2017)
17. Meshry, M., Goldman, D.B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., Martin-Brualla, R.: Neural rerendering in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 6878–6887
18. Yin, X., Wei, H., Wang, X., Chen, Q., et al.: Novel view synthesis for large-scale scene using adversarial loss. arXiv preprint arXiv:1802.07064 (2018)
19. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. In: ACM Transactions on Graphics. (2018)
20. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. In: CVPR. (2019)
21. Sitzmann, V., Zollhofer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-strcuture-aware neural scene representations. In: NeurIPS. (2019)
22. Thies, J., Zollhöfer, M., , Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. In: ACM Transactions on Graphics. (2019)
23. Srinivasan, P.P., Tucker, R., Barron, J.T., Ramamoorthi, R., Ng, R., Snavely, N.: Pushing the boundaries of view extrapolation with multiplane images. In: CVPR. (2019)
24. Thies, J., Zollhöfer, M., Theobalt, C., Stamminger, M., Nießner, M.: Ignor: Image-guided neural object rendering. In: arXiv. (2018)
25. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings video demo. https://youtu.be/-Vto65Yxt8s?t=228 (2019)
26. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. (2017)
27. Kar, A., Hane, C., Malik, J.: Learning a multi-view stereo machine. In: NIPS. (2017)
28. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Transactions on Graphics (TOG) **35** (2016) 1–10
29. Yang, J., Reed, S.E., Yang, M.H., Lee, H.: Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In: Advances in Neural Information Processing Systems. (2015) 1099–1107
30. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Interpretable transformations with encoder-decoder networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 5726–5735
31. Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: Hologan: Unsupervised learning of 3d representations from natural images. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7588–7597
32. Xu, X., Chen, Y.C., Jia, J.: View independent generative adversarial network for novel view synthesis. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7791–7800
33. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
34. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An invitation to 3-d vision: from images to geometric models. Volume 26. Springer Science & Business Media (2012)
35. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. CoRR **abs/1511.05952** (2015)

36. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. (2015) 945–953
37. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop. (2017)
38. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
39. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Single-view to multi-view: Reconstructing unseen views with a convolutional network. arXiv preprint arXiv:1511.06702 **6** (2015)
40. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 1125–1134
41. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM Transactions on Graphics (TOG) **38** (2019) 65
42. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Gated feedback recurrent neural networks. In: International Conference on Machine Learning. (2015) 2067–2075
43. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9** (1997) 1735–1780
44. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2790–2799
45. Papyan, V., Elad, M.: Multi-scale patch-based image restoration. IEEE Transactions on image processing **25** (2015) 249–261