CodeNeRF: Disentangled Neural Radiance Fields for Object Categories

Wonbong Jang Lourdes Agapito
Department of Computer Science
University College London
{ucabwja,l.agapito}@ucl.ac.uk

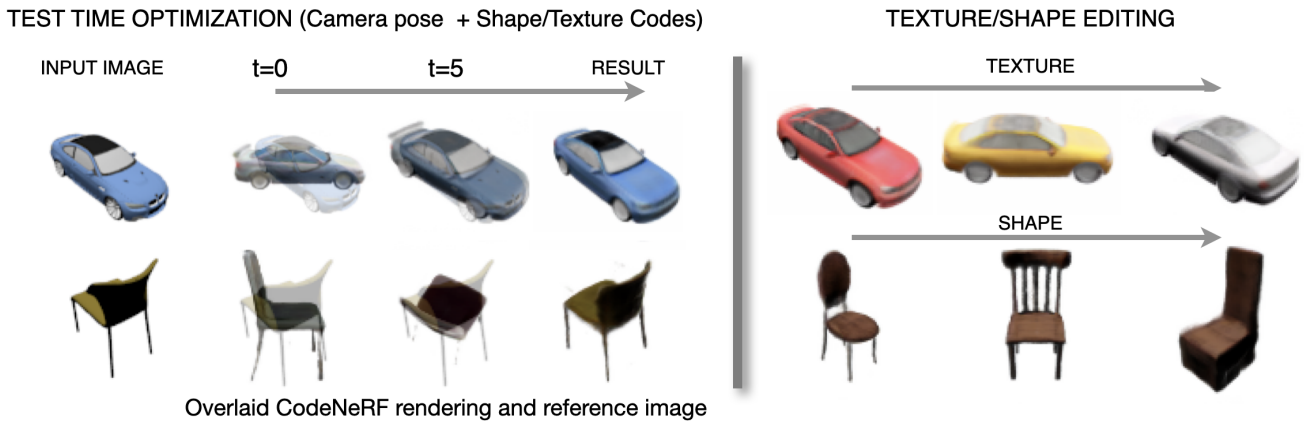


Figure 1: CodeNeRF represents variations of shape and texture across an object class. **Left**: At test time, given a single input image, the trained model can be used to jointly optimize camera viewpoint and shape/texture latent codes. **Right**: Object shapes, textures and viewpoints can later be edited simply by varying the latent codes, offering full control over synthesis.

Abstract

CodeNeRF is an implicit 3D neural representation that learns the variation of object shapes and textures across a category and can be trained, from a set of posed images, to synthesize novel views of unseen objects. Unlike the original NeRF, which is scene specific, CodeNeRF learns to disentangle shape and texture by learning separate embeddings. At test time, given a single unposed image of an unseen object, CodeNeRF jointly estimates camera viewpoint, and shape and appearance codes via optimization. Unseen objects can be reconstructed from a single image, and then rendered from new viewpoints or their shape and texture edited by varying the latent codes. We conduct experiments on the SRN benchmark, which show that CodeNeRF generalises well to unseen objects and achieves on-par performance with methods that require known camera pose at test time. Our results on real-world images demonstrate that CodeNeRF can bridge the sim-to-real gap. Project page: https://github.com/wayne1123/code-nerf

1. Introduction

Synthesizing novel views of unseen objects given a sparse set of input views or even a single image is a long-standing problem in the fields of computer vision and graphics. Synthesis methods require both an accurate representation of the 3D geometry and appearance of objects, and the ability to offer control over changes in viewpoint, shape or texture to render different objects of the same category.

Traditionally, discrete scene representations have been used, such as meshes or voxel grids that store geometry and appearance information either explicitly or via learnt neural features [23]. However, their discrete nature limits their representation power and resolution. With the recent introduction of scene representation networks (SRN), Sitzmann *et al*. [24] propose to learn a continuous function that map 3D locations to features of scene properties. Crucially, SRN does not require 3D supervision and can be trained end-to-end with a differentiable ray marcher that renders the feature-based representation into a set of 2D images. While SRN allows generalization to unseen objects, accurate cam-

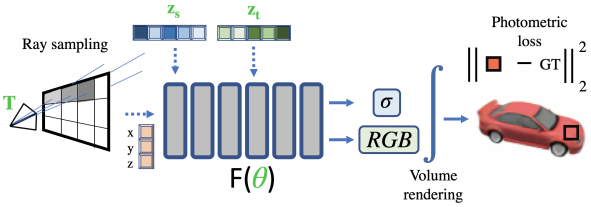


Figure 2: CodeNeRF disentangles geometry, appearance and viewpoint by learning separate embeddings for shape and texture and a fully connected network F(θ) that maps 3D locations and ray directions to density and RGB values.

	Deep SDF	SRN	NeRF	Pixel NeRF	CodeNeRF
Learned Prior	✓	✓	✗	✓	✓
Supervision	3D	2D	2D	2D	2D
Encoder	✗	✗	✗	✓	✗
Pose Optimization	-	✗	✗	✗	✓
Disentangled Shape & Texture	✗	✗	✗	✗	✓

Table 1: Related prior work on neural scene representations.

era pose estimates are required at inference time, and shape and appearance are represented in an entangled manner.

Mildenhall *et al*. [15] extended the neural representation to store volume density and view-dependent radiance values, enabling highly photo-realistic new view synthesis of complex real-world scenes that capture viewpoint dependent effects. New view synthesis is done by querying the network at each specific spatial location and viewing direction, followed by classic volume rendering to output image pixel intensities. While the quality of synthesized images is impressive, it requires a large number of images, and must be optimized for each new scene independently.

Neural representations have also been used to learn deformation priors that encode the variation of object shapes across semantic categories using direct 3D supervision [7, 2, 17, 14]. DeepSDF [17] is an auto-decoder architecture that jointly learns a latent embedding and the weights of a fully connected network that maps 3D coordinates to signed distance values. This representation uses test time optimization to estimate the shape code associated with a new unseen object. While DeepSDF [17] has been an extremely popular representation, successfully used as a shape prior to drive 3D reconstruction of object categories from multiple images [21, 20], its training requires 3D supervision and cannot be learnt from 2D images only.

Contributions: We present **CodeNeRF**, a novel 3D-aware representation for object categories that learns to disentangle shape and texture. During training, CodeNeRF takes a set of posed input images and simultaneously learns different latent embeddings for shape and texture, and the weights of a multi-layer perceptron to predict volume density and view-dependent radiance for each 3D point by enforcing multi-view photometric consistency. At inference, given a single unposed reference image of an unseen object, CodeNeRF optimizes shape and texture codes as well as camera pose. Our disentangled representation provides full control over the synthesis task, enabling explicit editing of object shape and texture simply by modifying the respective latent codes (see Fig 1). We demonstrate single view reconstruction, novel view synthesis and shape/texture editing on the SRN benchmark and real-world images.

While our work takes inspiration from other continuous neural scene representations [15, 24], it addresses many of their limitations. Unlike NeRF [15], CodeNeRF is not scene specific and can model the variations of shape and appearance across an object class. In contrast to SRN [24], CodeNeRF disentangles geometry and appearance offering explicit control over shape and texture for synthesis tasks. Unlike both, CodeNeRF does not require knowledge of camera pose at test time, estimating it via optimization. Inspired by DeepSDF [17], we adopt an auto-decoder architecture but depart significantly as we only require 2D supervision and can disentangle shape and texture variations across an object category. See Fig 2 for an overview of CodeNeRF.

也就是 camera extrinsic

2. Related Work

We focus our literature review on learning based approaches to 3D reconstruction and new view synthesis.

3D Reconstruction from 3D supervision Voxel-based methods were amongst the first [34, 3, 32] to propose learned representations for 3D reconstruction with 3D supervision. However, volumetric representations are fundamentally limited in terms of representation power and resolution and fail to capture surface details. Although point clouds are a common alternative to voxels, their unstructured nature and permutation-invariance make them difficult to handle using cnn architectures. PointNet [18] offered a unified architecture able to deal with unordered point sets via maxpooling, that may be used for multiple downstream learning tasks. Mesh-based representations have also been used for 3D shape estimation given 3D supervision [7] or applying graph convolutions [29, 6]. However, these methods still fail to capture fine details. DeepSDF [17] pioneered a new popular direction to represent 3D geometry via a continuous mapping from 3D points to an implicit representation of shape. DeepSDF jointly learns the weights of a fully connected network that maps 3D coordinates to signed distance values, and a latent embedding. While in principle DeepSDF can represent shapes at arbitrary resolutions without increasing its memory footprint, its training requires 3D supervision. Other concurrent approaches used

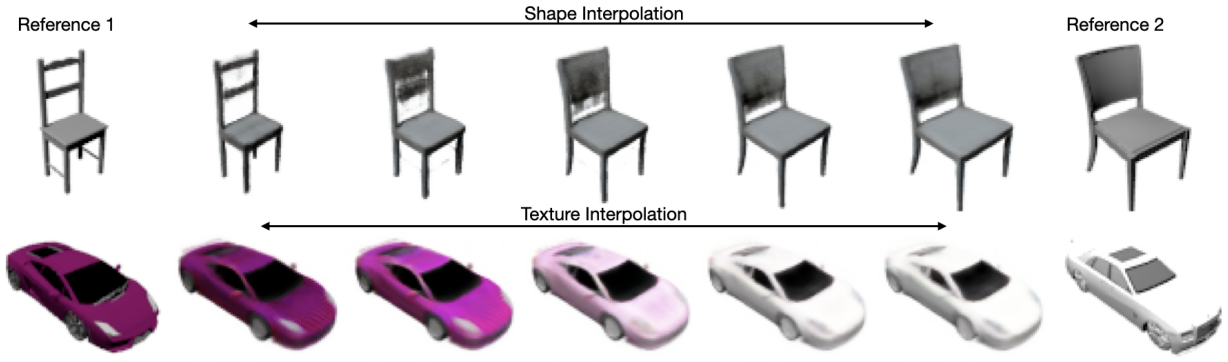


Figure 3: We illustrate the ability of CodeNeRF to disentangle geometry and appearance by synthesizing new shapes or textures via latent interpolation. We show two reference images (left and right). CodeNeRF first estimates the camera pose and shape+texture codes for these images and then interpolates between them.

continuous implicit occupancy networks [14, 2] but also require 3D annotations. A CNN-based encoder to extract local features to provide conditioning priors was used in [35]. However, all these methods require heavy pre-processing of the training data to compute the SDF of query points. In addition, if a mesh is required during optimization for rendering, marching cubes must be used, which is not easily differentiable [11]. Mesh-based, SDF or occupancy representations pre-learnt with 3D supervision [14, 2, 35] can be combined with a CNN encoder to perform single view reconstruction. However, as noted in [27], these models often perform recognition instead of 3D reconstruction.

3D Reconstruction from 2D supervision The emergence of differentiable renderers [8, 10, 19] allowed to obtain gradients through the rendering process, enabling them to be embedded into end-to-end learning frameworks. DeepSDF was used as a deep pre-learnt shape prior for 3D shape optimization from multiple images [21] requiring 2D supervision only. MeshSDF [20] circumvents the non-differentiability of marching cubes and uses 2D supervision based on a silhouette loss. A mesh-based representation is optimized from 2D images in [12]. The disadvantage of many of these methods is that the learning of shape priors is decoupled from the shape estimation.

Neural 3D representations DVR [16] proposes a differentiable rendering formulation for implicit shape and texture representations that can be learnt from multi-view images and object masks only, without the need for 3D supervision. Sitzmann *et al*. [24] introduced SRN (Scene Representation Networks) a 3D-aware representation that learns a scene prior via a continuous mapping of 3D spatial locations to features of scene properties. Similar to our approach, once trained, the model can be used as a learned scene prior via test-time optimization. However, unlike CodeNeRF, test-time optimization for SRN requires known absolute camera poses, which can be seen as strong supervision.

Neural Radiance Fields (NeRF) for new view synthesis NeRF [15] extended the neural representation to allow to capture viewpoint varying image properties by storing volume density and view-dependent radiance values which enabled photorealistic new view synthesis of complex real-world scenes. Many new extensions to NeRF have been recently proposed, some concurrent to ours. GRAF [22] introduces the idea of *conditional* radiance fields into the generative adversarial network framework, to convert it into a 3D aware generative model. In addition to new viewpoint synthesis, similarly to ours, their approach allows to modify shape and appearance of the generated objects via shape and texture latent embeddings. However, unlike ours, their method is a purely generative model, trained on an adversarial loss over patches. As such, it cannot estimate shape, texture or camera pose given an input image so cannot be used for single view reconstruction. PixelNeRF [37] can generalise to multiple scenes by using an image encoder to condition the neural radiance field on image features and does not require test-time optimization. However, unlike our approach, there is no disentanglement of geometry and appearance and therefore no control over shape or texture editing. iNeRF [36] uses a pre-trained NeRF to optimize the camera pose while NeRF−−[31] jointly estimates the neural representation and camera intrinsic and extrinsic parameters but both are scene specific.

3. Methodology

Given a training set of N images depicting M objects across a semantic class, along with their respective camera intrinsics and pose parameters $\mathcal{V} = \{\mathcal{I}_i, \mathrm{K}_i, \mathrm{T}_i\}_{i=1}^{\mathrm{N}}$, CodeNeRF jointly learns the weights of a multi-layer perceptron F_Θ that encapsulates the geometry and appearance across the observed objects, and separate latent embeddings

Training 的時候知道
V。
而 V 三個分別
I 表示 image
K 表示 intrinsic
T 表示 pose (extrinsic)

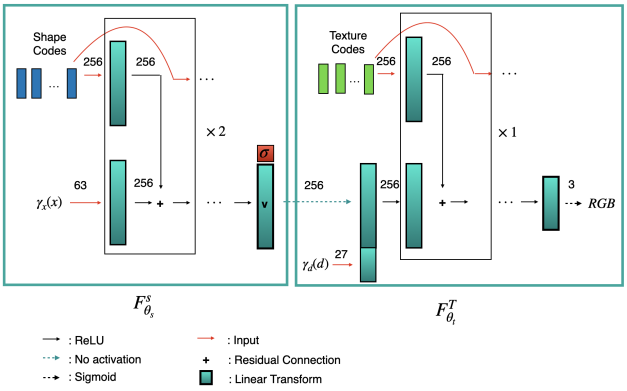


Figure 4: **CodeNeRF architecture.** $\gamma_x(\mathbf{x})$ and $\gamma_d(\mathbf{d})$ are positional encodings for 3D point coordinates and viewing directions respectively. Volume density σ does not depend on texture codes $\mathbf{z_t}$

$\{\mathbf{z_s}^j, \mathbf{z_t}^j\}_{j=1}^{\mathrm{M}} \in \mathbb{R}^{256}$ that disentangle shape and appearance respectively. F_Θ is a disentangled neural radiance field that takes shape and texture codes as inputs and maps scene coordinates $\mathbf{x}$ and viewing directions $\mathbf{d}$ to their corresponding volume density σ and RGB colour values $\mathbf{c}$:

$$\mathrm{F}_\Theta : (\gamma_x(\mathbf{x}), \gamma_d(\mathbf{d}), \mathbf{z_s}, \mathbf{z_t}) \rightarrow (\sigma, \mathbf{c}) \tag{1}$$

Following [15] we use positional encoding $\gamma(\cdot)$ for both scene coordinates $\mathbf{x}$ and viewing directions $\mathbf{d}$ to capture high frequency details. We employ 10 frequencies for $\mathbf{x}$ and 4 for $\mathbf{d}$. We design the architecture of the decoder to take advantage of the fact that the volume density σ depends only on the 3D point $\mathbf{x}$ and shape code $\mathbf{z_s}$, while the RGB color depends in addition on the viewing direction $\mathbf{x}$ and texture code $\mathbf{z_t}$. The first layers of the MLP F^s_Θ map the input 3D coordinate $\gamma(\mathbf{x})$ and shape code $\mathbf{z_s}$ to the volume density σ and an intermediate feature vector $\mathbf{v} \in \mathbb{R}^{256}$. The second part of the network F^t_Θ takes $\mathbf{v}$ and $\mathbf{z_t}$ as input and outputs the RGB colour as shown in Fig. 4.

$$\begin{aligned} \mathrm{F}^s_{\Theta_s} &: (\gamma_\mathrm{x}(\mathbf{x}), \mathbf{z_s}) \rightarrow (\sigma, \mathbf{v}) \\ \mathrm{F}^t_{\Theta_t} &: (\mathbf{v}, \gamma_\mathrm{d}(\mathbf{d}), \mathbf{z_t}) \rightarrow (\mathbf{c}) \\ \mathrm{F}_\Theta &: \mathrm{F}^s_{\Theta_s} \circ \mathrm{F}^t_{\Theta_t} \end{aligned} \tag{2}$$

Conditioning on Learned Shape and Texture Embeddings Inspired by SRN [24] and DeepSDF [17], CodeNeRF adopts an auto-decoder architecture with decoupled shape and texture embedding spaces. During training, embedding vectors $\{\mathbf{z_s}^j, \mathbf{z_t}^j\}_{j=1}^{\mathrm{M}}$ are optimized jointly with the parameters of the network F_Θ. Unlike DeepSDF [17] the shape and texture embeddings can be trained end-to-end from images only, without requiring any 3D supervision. In contrast to SRNs [24] shape and texture embeddings are decoupled and can be used as separate controls for editing/synthesis. Figs. 1, 3 and 9 show how the embeddings can be used as conditioning to render smooth interpolations between different shapes/textures in the latent spaces. Jointly, the shape/texture embeddings and the decoder network F_θ act as learned priors that model shape deformations and texture variations across a semantic category. Once the model is trained, given a single input image, it can be used for one-shot reconstruction of an unseen object, via a test-time optimization that estimates the corresponding shape and texture codes $\{\mathbf{z_s}^j, \mathbf{z_t}^j\}$ and even the camera pose (see Sec. 3.2). We show that CodeNeRF generalises well to render novel viewpoints, demonstrating that the learnt priors help to complete unobserved geometry and texture patterns (see Figs. 10 and 13).

Volume Rendering We follow [15] and use classical volumetric rendering techniques to render the color of image pixels by aggregating the color and the occupancy density at N uniform samples from the near to far bounds for the camera ray $r(t) = c + t\mathbf{d}$ traced through each pixel. The estimated color $C(r)$ of the pixel can be expressed as:

$$\hat{C}(r) = \sum_{i=1}^{\mathrm{N}} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \tag{3}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples and $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$. This expression is fully differentiable, and reduces to the traditional alpha compositing with alpha values. NeRF [15] adopts a 2 stage approach, a coarse and a fine network, with the fine network using importance sampling based on the values of T_i from the coarse network to sample closer to the surface.

3.1. Training CodeNeRF

For each image in the training set $\mathcal{V} = \{\mathcal{I}_i, \mathrm{K}_i, \mathrm{T}_i\}_{i=1}^{\mathrm{N}}$, at each training iteration we sample a batch of 4094 rays using the intrinsic and extrinsic parameters. We then sample 64 points along each ray and feed them to the MLP F_Θ together with the current estimates of shape and texture codes. F_Θ provides the occupancy density σ and the RGB color $\mathbf{c}$ and volume rendering is used to aggregate colors along each ray. Training is supervised using the photometric loss along with a regularization loss used as a prior over the latent vectors (see Eq. (4)). We train the model with AdamW [13] optimizer with initial learning rates 1e-4 for the network parameters and 1e-3 for the latent vectors.

$$\begin{aligned} \mathcal{L}(\Theta, \{\mathbf{z_s}^i, \mathbf{z_t}^i\}^{\mathrm{M}}) &= \sum_{r \in \mathcal{R}} ||\hat{C(r)} - C(r)||_2^2 \\ \min_{\Theta, \{\mathbf{z_s}^i, \mathbf{z_t}^i\}^{\mathrm{M}}} \mathcal{L}(\Theta, \{\mathbf{z_s}^i, \mathbf{z_t}^i\}) &+ \frac{1}{\nu^2}(||\mathbf{z_s}^i||_2^2 + ||\mathbf{z_t}^i||_2^2) \end{aligned} \tag{4}$$

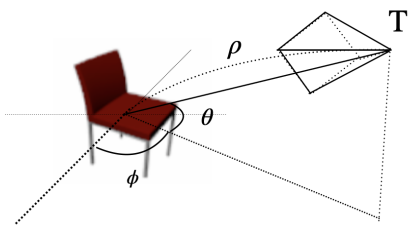


Figure 5: Camera viewpoint parametrization. At test time, CodeNeRF optimizes camera pose jointly with latent codes.

3.2. Inference Optimization

At inference time, given a single input image of an unseen object, the trained model can be used to optimize the latent vectors by minimizing the photometric loss between the rendered and observed pixels (Eq. 5), while keeping the weights of the neural network fixed. Moreover, unlike previous methods [24, 37] we show that CodeNeRF does not require known camera pose and its parameters can also be optimized jointly with the latent codes.

Camera Pose Optimization: CodeNeRF backprojects pixels into 3D rays using the camera to world transformation matrix. Similarly to NeRF [15], we require known camera matrices to train the model. However, once the model has been learnt, at inference time, unlike others [24, 15, 37], we do not require known camera viewpoint and we optimize the rotation and translation parameters jointly with the latent embedding vectors, by obtaining the gradients over the input points and ray directions, in a similar way to a differentiable renderer. Similarly to other object-category 3D reconstruction methods, we assume a simplified camera model, where the world coordinate system is centred on the 3D object, the camera faces its origin, and the up vector is aligned with the z axis. In this case, $\mathbf{R}$ and $\mathbf{t}$ can be parametrized via azimuth ϕ, elevation θ and distance to the camera ρ, as shown in Fig. 5. The camera to world transformation matrix can be written as $\mathbf{T}_{cw} = \begin{pmatrix} \mathbf{R}^T & \mathbf{p} \\ 0 & 1 \end{pmatrix}$ where:

$$\mathbf{R} = \begin{pmatrix} -\sin\phi & \cos\phi & 0 \\ -\sin\theta\cos\phi & -\sin\theta\sin\phi & \cos\theta \\ \cos\theta\cos\phi & \cos\theta\sin\phi & \sin\theta \end{pmatrix}$$

and $\mathbf{p} = (\rho\cos\theta\cos\phi, \rho\cos\theta\sin\phi, \rho\sin\theta)$ is the camera location. At inference time, given an reference image $\mathcal{I}_i$, the camera pose and the sampled rays γ_i can be expressed in terms of the camera parameters azimuth (ϕ, θ, ρ) and optimized via using the gradients.

Test time optimization: We minimize the photometric loss (5) jointly with respect to shape and texture codes and camera parameters (fixing the decoder parameters Θ), using the AdamW optimizer.

$$\min_{\mathbf{z_s}^i, \mathbf{z_t}^i, \rho_i, \theta_i, \phi_i} \mathcal{L}(\mathbf{z_s}^i, \mathbf{z_t}^i, \rho_i, \theta_i, \phi_i) + \frac{1}{\nu^2}(||\mathbf{z_s}^i||_2^2 + ||\mathbf{z_t}^i||_2^2) \tag{5}$$

Latent vectors are initialized with the mean vector of the trained embeddings, and an initial learning rate of 1e-2 is used. The initial learning rates for the camera parameters are 1e-2, 1e-1 and 1e-1 respectively.

Enforcing disentanglement: To illustrate the link between good architecture choices and disentanglement, we trained two alternative models (M1/M2). In M1, $\gamma_d(d)$ is given as input to the first layer of $F^s_{\theta_s}$ jointly with 3D point position $\gamma_x(x)$ (similarly to PixelNeRF). In M2, z_s and z_t are given as input to $F^s_{\theta_s}$, which amounts to using a single embedding. We then perform texture editing keeping the shape code fixed. Fig. 6 shows how only CodeNeRF fully disambiguates shape and texture. M1 fails to synthesise the correct texture while M2 has unwanted changes in shape.

Figure 6: Shape/Texture disentanglement: Alternative architectures fail to disentangle shape and texture.

4. Experimental Evaluation

We train our model on the ShapeNet-SRN dataset of ShapeNet [1] renderings for two object categories: cars (3514) and chairs (6591), created by Sitzmann *et al*. [24]. We train a different model for each category using the predefined train/test split and conduct a quantitative evaluation on novel view synthesis and few shot reconstruction tasks.

4.1. Quantitative Evaluation on SRN Benchmark

The ShapeNet-SRN benchmark [24] provides 251 test images on an Archimedeal spiral per object in the test set. We follow the evaluation protocols provided by SRN [24].

Baselines We compare quantitatively on the ShapeNet benchmark [24] with SRN [24], ENR [4], PixelNeRF [37], and other current state of the art methods, on the tasks of one-view and two-view 2D-supervised reconstruction. Note that competing approaches require known camera pose at inference time. We compare with three variants of our approach: CodeNeRF (GT pose) assumes known camera pose at test time, CodeNeRF is our full approach with joint optimization of camera pose and shape/texture codes (299 iterations) and CodeNeRF (− outliers), which is CodeNeRF but without counting results that resulted in an outlier (rotation error above 5° or translation error higer than 3%).

把預測不好的 pose 給移除

	1-view		2-view	
	PSNR	SSIM	PSNR	SSIM
Chairs				
GRF [28]	21.25	0.86	22.65	0.88
TCO [26]	21.27	0.88	21.33	0.88
dGQN [5]	21.59	0.87	22.36	0.89
ENR [4]	22.83	-	-	-
SRN [24]	22.89	0.89	24.48	0.92
PixelNeRF [37]	**23.72**	**0.91**	**26.20**	**0.94**
CodeNeRF (GT pose)	23.66	0.90	25.63	0.91
CodeNeRF	22.39	0.87	-	-
CodeNeRF (− outliers)	23.11	0.89	-	-
Cars				
SRN [24]	22.25	0.89	24.84	0.92
ENR [4]	22.26	-	-	-
PixelNeRF [37]	23.17	0.90	25.66	**0.94**
CodeNeRF (GT pose)	**23.80**	**0.91**	**25.71**	0.93
CodeNeRF	22.73	0.89	-	-
CodeNeRF (− outliers)	23.17	0.90	-	-

Table 2: Quantitative evaluation on ShapeNet-SRN

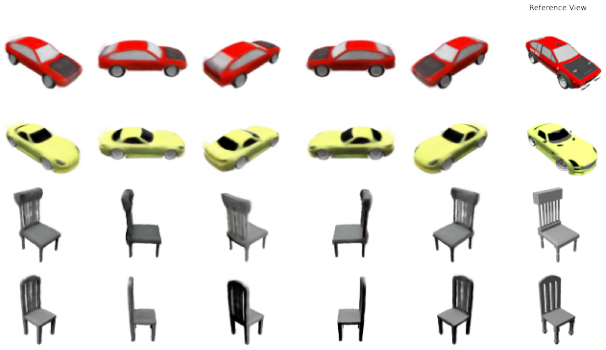


Figure 7: Qualitative results of novel view synthesis of unseen objects from a single input image with CodeNeRF (GT pose), a variant of CodeNeRF which assumes known camera pose at test time. Rightmost column shows input view.

Metrics We use PSNR(Peak Signal to Noise Ratio) and SSIM(Structural Similarity Index Measure) [30] to evaluate the quality of the rendered images.

Synthesis Results Table 2, shows a full comparison of the results on 1-view and 2-view reconstruction. CodeNeRF (GT pose) – which assumes known camera pose, for a fairer comparison with its competitors that need it as input – outperforms PixelNeRF for cars and comes very close on chairs. Even when CodeNeRF estimates the unknown pose at test time, along with the latent codes, the performance does not degrade much. When pose estimation outliers are removed the performance is close to PixelNeRF. Qualitative results for 1-view reconstruction with CodeNeRF (GT pose) are shown in Fig. 7 while Fig. 8 shows results for CodeNeRF with camera pose+latent codes optimization. Clearly, rendered images become sharper when the ground truth pose is used, but as Fig. 8 shows, even when pose and latent codes are initialized far from the ground truth, CodeNeRF converges to good estimates.

Figure 8: Test-time joint optimization of camera pose and latent codes: Evolution of estimated renderings from initialization (left), through intermediate optimization iterations $5/10/50$, to final result after 299 iterations.

Editing Novel Shapes and Textures After the test-time optimization has taken place, CodeNeRF provides full control over the synthesis process as object shapes and textures can be edited simply by varying the corresponding latent codes. This is possible given that viewpoint, shape and texture are fully disentangled in our representation. As shown in Fig. 9, we can easily fix one while changing the other.

Evaluation of Camera Viewpoint Estimation: At test time, CodeNeRF optimizes jointly the camera pose and shape/pose latent codes. We run a quantitative evaluation of the performance of the camera pose estimation on the ShapeNet-SRN dataset. Fig. 11 (top) shows histograms with the distribution of pose estimation errors. It is interesting to note that the initializations we used for the camera pose parameters (shown in green) had rotation errors above 30° in 100% of cases and were on average 80° away from GT. Despite this, our optimization converged in 85% of cases. The table in Fig. 11 (below) shows the numerical errors between the estimated and ground-truth camera poses in terms of the percentage of rotation estimates with errors below 5° and 10°, and the percentage of translation estimates with relative errors below 3% and 5%. Fig. 8 shows the initialization and illustrates the evolution of the intermediate results obtained through the optimization (iterations 5/10/50), and final result after 299 iterations. Even when the initial estimate is far away from the ground truth, the optimization converges to good solutions.

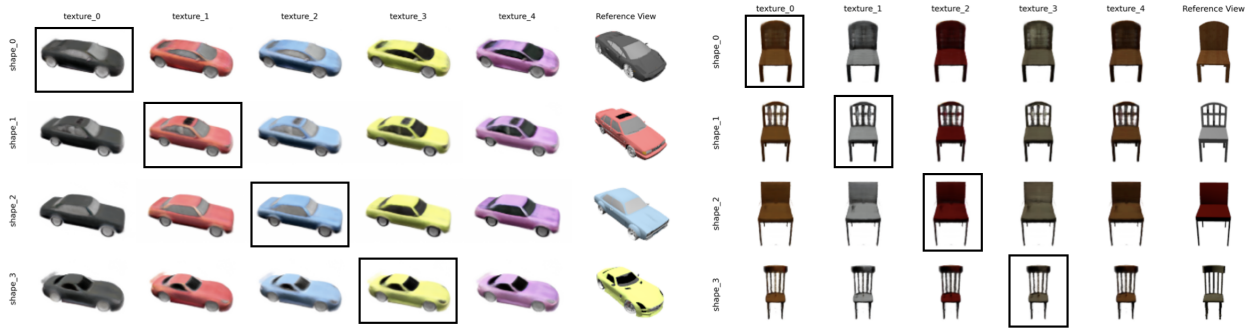


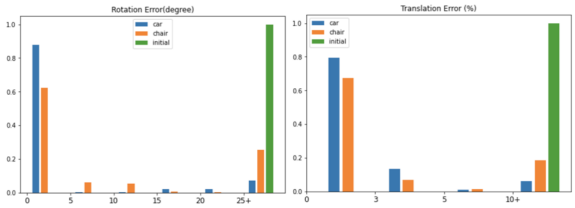
Figure 9: Novel Shape/Texture/Pose Synthesis. Highlighted results show the renderings with shape and texture codes corresponding to the reference views (obtained via optimization). Others show results of varying shapes and textures simply by editing corresponding latent codes.

Figure 10: Qualitative results on real-world datasets (Stanford-Car [9] and Pix3D [25]) of CodeNeRF with joint optimization of camera pose+latent codes. Evolution of estimated renderings throughout the optimization from initialization through iterations $5/10/50$ and final result after 299 iterations. Rightmost column shows input image.

train 在 ShapeNet-SRN 上，但使用真實 dataset 做 test。
camera intrinsic 假設跟 ShapeNet-SRN 一樣，pose(extrinsic) 就可以自己預測

4.2. Qualitative Comparisons

Fig. 12 shows examples comparing CodeNeRF with PixelNeRF [37] and SRN [24] on new view synthesis of objects from the ShapeNet-SRN test set. Since PixelNeRF uses a pre-trained CNN to extract features, it renders sharper images when the target view is close to the input view. SRN works better on occluded regions than PixelNeRF as it learns a latent embedding that acts as a priors. CodeNeRF achieves shape and texture disentanglement maintaining geometric and appearance consistency in occluded regions. Only CodeNeRF can synthesise the correct combination of shape and texture for the purple chair.



		Rotation Error		Translation Error	
		5°	10°	3%	5%
Ours	Cars	87.8%	88.1%	79.3%	92.7%
	Chairs	62.3 %	68.3 %	67.3 %	74.0%

Figure 11: Pose estimation evaluation. Error distribution histograms: rotation (top-left), translation (top-right). Initialization error in green. Numerical evaluation (below).

4.3. Results on Real World Datasets

To understand if our model, trained only on synthetic renderings from ShapeNet-SRN, can generalise well to real-world images, we perform an evaluation on two datasets: Stanford-Car dataset [9] and Pix3D [25]. We perform test time optimization given a single input image to estimate jointly camera pose+latent codes. We then render the representation from unseen viewpoints.

Pre-processing Similarly to PixelNeRF [37], we use Detectron2 [33] on the Stanford-Car dataset to infer masks, then apply Gaussian blur and downscale the images to 128×128. For the real chairs in Pix3D [25], we carve out the object with the provided ground-truth mask before downscaling.

Optimization of Camera Pose and Latent Vectors We show results of one-shot reconstruction given a single in-

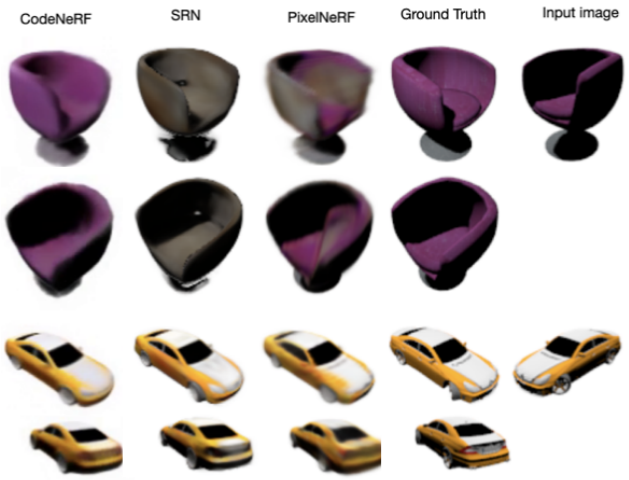


Figure 12: Qualitative comparisons with SRN [24] and PixelNeRF [37] (results courtesy of A. Yu and V. Sitzmann).

Figure 13: Qualitative evaluation of novel view synthesis on images from real-world datasets (Stanford-Car [9] and Pix3D [25]). Input image is shown on the rightmost column. Camera pose is estimated jointly with latent codes.

put image with unknown camera pose. Since the camera intrinsics are unknown, we assume the same focal length as in ShapeNet-SRN. Fig. 10 shows the initialization and the intermediate results through different iterations of the optimization. The optimization provides convincing final renderings even when initialised from far away.

Shape, Texture and Viewpoint Editing Following one-shot optimization, we show renderings of the reconstructed objects from novel viewpoints (see Fig. 13). Fig. 14 illustrates results after editing shape and texture latent vectors. The distinctive style of each object is preserved after shape or texture transfer. The shape and texture priors learned by CodeNeRF help to complete object shape and texture information that was not present in the input image.

Explicit mesh reconstruction CodeNeRF implicitly represents the 3D structure of reconstructed objects. To obtain an

Figure 14: Shape and texture editing on real images from the Stanford-Cars and Pix3D datasets.

explicit mesh representation, we feed the centre of of 256^3 voxels to the MLP, and obtain their volume density values σ. Marching cubes can then be used to find the mesh vertices. Casting rays along vertex normals we can associating colors to the vertices. Fig. 15 shows the input image and the reconstructed meshes for a real image from the Stanford-Car dataset and a synthetic car from ShapeNet.

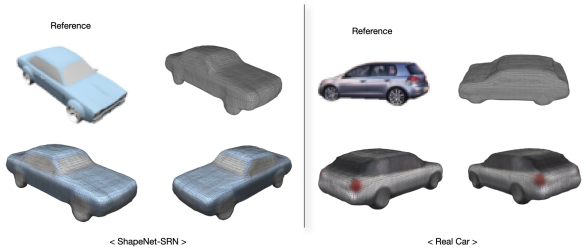


Figure 15: **Reconstructing 3D meshes:** CodeNeRF allows 3D reconstruction of a mesh from a single input image.

5. Conclusion

We have presented CodeNeRF, a neural radiance field that learns to disentangle shape and appearance by jointly learning separate latent embeddings for shape and texture, and an MLP that maps continuous input locations and ray directions to density and colour information. At inference time, given a single input image CodeNeRF can estimate its associated camera pose and latent codes. We show extensive results of one-shot reconstruction on the SRN benchmark and on real world images (Stanford-Cars and Pix3D) that demonstrate that CodeNeRF learns powerful scene priors that enable full control over the synthesis process.

Acknowledgements: Research presented here has been supported by funding from Cisco to the UCL AI Centre.

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] Zhiqin Chen. *IM-NET: Learning implicit fields for generative shape modeling*. PhD thesis, Applied Sciences: School of Computing Science, 2019.

[3] 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 *European conference on computer vision*, pages 628–644. Springer, 2016.

[4] Emilien Dupont, Miguel Bautista Martin, Alex Colburn, Aditya Sankar, Josh Susskind, and Qi Shan. Equivariant neural rendering. In *International Conference on Machine Learning*, pages 2761–2770. PMLR, 2020.

[5] 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.

[6] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9784–9794, 2019.

[7] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[8] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[10] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[11] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[12] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 969–978, 2019.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[14] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[16] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[17] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[19] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[20] Edoardo Remelli, Artem Lukoianov, Stephan R. Richter, Benoit Guillard, Timur M. Bagautdinov, P. Baqué, and P. Fua. Meshsdf: Differentiable iso-surface extraction. *ArXiv*, abs/2006.03997, 2020.

[21] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, et al. Frodo: From detections to 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14720–14729, 2020.

[22] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020.

[23] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[24] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020.

[25] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] 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.

[27] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do

single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.

[28] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020.

[29] 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 (ECCV)*, pages 52–67, 2018.

[30] 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.

[31] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf $--$: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[32] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[33] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[34] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[35] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502, 2019.

[36] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. *arXiv preprint arXiv:2012.05877*, 2020.

[37] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.