Few-View Object Reconstruction with Unknown Categories and Camera Poses

Hanwen Jiang Zhenyu Jiang Kristen Grauman Yuke Zhu
Department of Computer Science, The University of Texas at Austin
{hwjiang,zhenyu,grauman,yukez}@cs.utexas.edu

Abstract

While object reconstruction has made great strides in recent years, current methods typically require densely captured images and/or known camera poses, and generalize poorly to novel object categories. To step toward object reconstruction in the wild, this work explores reconstructing general real-world objects from a few images without known camera poses or object categories. The crux of our work is solving two fundamental 3D vision problems — shape reconstruction and pose estimation — in a unified approach. Our approach captures the synergies of these two problems: reliable camera pose estimation gives rise to accurate shape reconstruction, and the accurate reconstruction, in turn, induces robust correspondence between different views and facilitates pose estimation. Our method FORGE *predicts 3D features from each view and leverages them in conjunction with the input images to establish cross-view correspondence for estimating relative camera poses. The 3D features are then transformed by the estimated poses into a shared space and are fused into a neural radiance field. The reconstruction results are rendered by volume rendering techniques, enabling us to train the model without 3D shape ground-truth. Our experiments show that* FORGE *reliably reconstructs objects from five views. Our pose estimation method outperforms existing ones by a large margin. The reconstruction results under predicted poses are comparable to the ones using ground-truth poses. The performance on novel testing categories matches the results on categories seen during training. Project page: https://ut-austin-rpl.github.io/FORGE/*

1. Introduction

Reconstructing real-world objects through the lens of RGB cameras is crucial in AR/VR applications [3, 36], embodied AI [23, 46] and robotics [11, 59]. Conventional methods [15, 27, 41] rely on densely captured images for optimization-based reconstruction. The stringent requirement on dense inputs hinders the broad applicability of these methods. In contrast, few-view object reconstruction [61, 63] aims to quickly create 3D models from a few images of real-world objects, such as online product snapshots. Yet, existing work requires either 3D ground-truth supervision for training [7] or well-calibrated camera poses for inference [13, 63], and often restricts to seen object categories [61]. To move toward efficient capture of real-life objects, we aspire to develop a practical approach that performs general few-view object reconstruction with no reliance on object categories or camera poses.

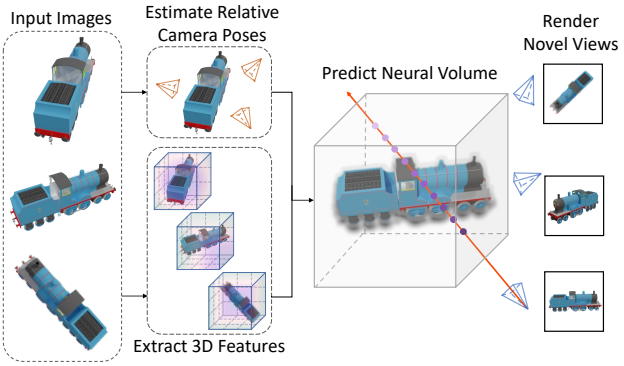


Figure 1. **Few-view object reconstruction.** FORGE reconstructs a 3D object from a few views without camera poses. It estimates relative camera poses between input views and extracts per-view 3D features. The features are fused based on the predicted poses to predict a neural volume, encoding the radiance field. We use volume rendering to generate the reconstruction results.

In this paper, we introduce FORGE (Few-view Object Reconstruction that GEneralizes), illustrated in Fig. 1. Departing from category-level methods that reconstruct objects in a category-specific canonical space [12, 53], FORGE encodes individual inputs into 3D features in their own camera spaces. It then estimates relative camera poses between input views and transforms the 3D features into a shared reconstruction space with the estimated poses. This design eliminates the need for a canonical reconstruction space of each category and enables FORGE to generalize across object categories. The transformed 3D features are subsequently aggregated into a neural volume. We follow NeRF [26] and use differentiable volume rendering to predict novel views from the neural volume. The model is supervised with a reconstruction loss between the rendered and raw input images,

	(1) Sparse views	(2) W/o poses	(3) Novel category	(4) Feed-forward	(5) W/o shape	(6) Explicit Recon.
COLMAP [41]	✗	✓	✓	✗	✓	✓
NeRF [26]	✗	✗	✓	✗	✓	✓
P-NeRF [63]	✓	✗	✓	✓	✓	✓
SRT [40]	✓	✓	✓	✓	✓	✗
FvOR [61]	✓	✓	✗	✗	✗	✓
FORGE	✓	✓	✓	✓	✓	✓

Table 1. **Model feature comparison.** (1) Model operates on a few sparse input views; (2) Ability to infer without camera poses of inputs; (3) Ability to perform reconstruction on novel category; (4) Fast, feed-forward reconstruction; (5) No need for 3D ground-truth of object shape; (6) Ability to produce explicit 3D reconstruction of point cloud, mesh or voxel.

without 3D supervision for object shape.

The foremost challenge is estimating relative camera poses between views. Existing works on camera pose estimation leverage correspondence of 2D images [22, 42, 64]. For few-view object reconstruction, drastic variation of camera poses impedes the establishment of robust 2D correspondence. To this end, we design a novel relative pose estimator with both 3D features and 2D images as input, exploiting the correspondence of the 3D features to eliminate reprojection ambiguity. Furthermore, to avoid compounding errors in pairwise relative pose estimation [17, 39], FORGE predicts poses based on the correspondences across all inputs. It thus benefits from the 3D correspondence and has a global understanding of the camera configurations.

FORGE exploits the synergies between shape reconstruction and pose estimation to improve performance on both. We first train the model with ground-truth camera poses. Using ground-truth camera poses encourages the model to learn 3D geometry priors for extracting consistent 3D features from each input view. We then learn the relative camera pose estimator, which builds 3D correspondence accurately in the well-established view-consistent 3D feature space.

To evaluate the generalization ability of FORGE, we design a new dataset with diverse object categories and harder camera settings compared to previous datasets [45, 57]. Our results show that FORGE outperforms prior art on both reconstruction and relative camera pose estimation by a large margin. The reconstruction quality under predicted poses matches the performance using ground-truth relative camera poses. Moreover, FORGE exhibits a strong generalization ability on novel object categories, almost on par with the performance on seen objects. Further, we can easily obtain accurate voxel reconstruction from the predicted neural volume, demonstrating the power of FORGE in understanding 3D geometry.

We highlight our contributions as follows: i) We develop a few-view object reconstruction approach that generalizes to novel object categories; ii) We design a new relative pose estimation model to handle large camera pose variations; iii) Our experiments demonstrate the synergies between shape reconstruction and pose estimation for improved quality on both tasks. We are committed to **release our code and dataset** for reproducibility and future research.

2. Related Work

Multi-View Reconstruction. Reconstructing objects and scenes from multi-view images has been a long-standing problem in computer vision [44]. Traditional methods, e.g., COLMAP [41], and their learning-based counterparts, e.g., DeepV2D [48], have already shown great success. However, these SLAM [4, 15, 48, 69], SfM [41, 47, 48] and RGB-D registration [2] approaches require dense-view inputs and smooth camera movement.

Another line of work aims at using only sparse or even a few views as input, where two main streams emerged. The first genre is pose-free. For example, 3D-R2N2 [7] and Pix2vox++ [58] aggregate 3D information from all inputs directly. SRT [40] builds a geometry-free method using a large transformer model. However, geometry-free methods are hard to generalize to unseen categories due to the absence of operations that aware of cross-view correspondence. The other approach is pose-aware, where camera poses are used for aligning features of each view into a common reconstruction space [13, 30, 31, 33, 54], and they benefit from using ground-truth camera poses. In this paper, we develop a pose-aware method that predicts both shape and relative camera poses in a unified manner without using ground-truth camera poses during inference.

Volumetric 3D and Neural Radiance Fields. 3D explicit volumetric representations, especially voxel grids, have been widely used for modeling objects [7, 8, 13, 28, 38, 50, 60] and scenes [16, 49]. In recent years, given the impressive results of implicit representations on 3D vision tasks [25, 32], NeRF [26] adopts an implicit neural radiance field for 3D volumetric representations. NeRF and its variants achieve a solid ability to model complex geometry and appearance [24, 34, 35, 43, 66]. Nonetheless, NeRF uses one global MLP for fitting each scene, which is challenging to optimize and suffers from limited generalization.

Two types of methods are proposed to solve the limitations of conventional NeRF methods. The first is integrating 2D image features into the MLP [37, 55, 63], which is sensitive to camera pose error due to the 3D-2D projection. The second is using a semi-implicit radiance field, where the radiance field is attached to voxel grids [20]. Furthermore, to make the radiance field more generalizable, some work [6, 62, 67] trained 3D encoders which directly predict voxel-based radiance fields from images. However, MVS-NeRF [6] is set to a fixed number of nearby views, ShelfSup [62] predicts radiance fields in the canonical space, and NeRFusion [67] requires ground-truth camera poses. In contrast, our model predicts the voxel-based radiance field from raw images, which can be fused using an arbitrary number

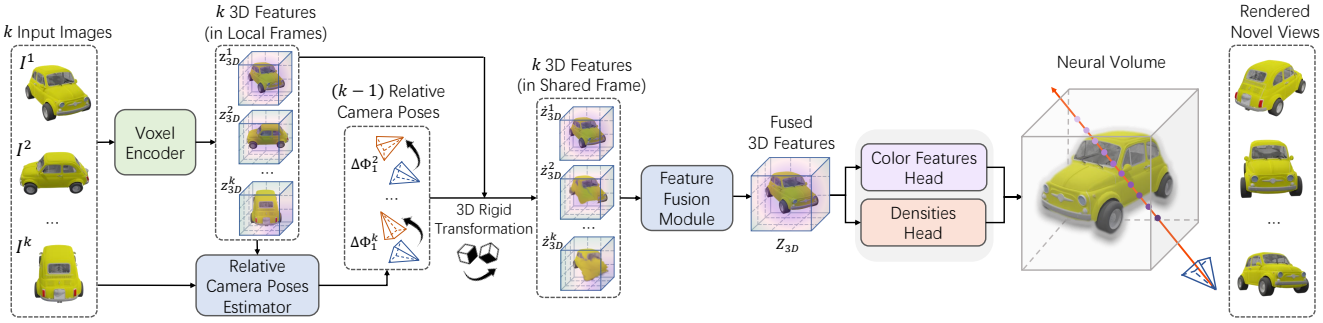


Figure 2. **Model overview.** FORGE extracts 3D voxel features from each view and uses the 3D features, along with 2D observations, to estimate relative camera poses. The 3D features in their corresponding cameras' frames are transformed into a shared reconstruction space using the rigid transformation computed by the relative camera poses. The features are fused to predict a neural volume that encodes the radiance field. We use volume rendering techniques to render the reconstruction results.

of views using predicted poses.

Reconstruction from Images without Poses. One drawback of pose-aware reconstruction methods using a volumetric representation is the requirement for accurate camera poses. Many works assume access to ground-truth camera poses [13,26,45], which limits their applicability. BARF [18] and NeRS [65] performed joint optimization on shape and pose. However, they still rely on highly accurate initial poses. FvOR [61] proposed a pose initialization module using dense point correspondence but requires 3D shape supervision.

Another line of work [17, 28, 51] takes advantage of the synergies between shape and pose. GRNN [51] trains a relative pose estimator and deploys it for predicting the poses during reconstruction. VideoAE [17] learns the relative camera poses and 3D representations under fully unsupervised learning by disentangling the shape and pose. However, these two works predict the relative camera poses from raw 2D views, making them hard to handle unseen categories, texture-less objects, and large pose variations due to the 2D ambiguity. Our method predicts relative camera poses in world coordinates. We use the predicted poses for performing cross-view fusion, linking shape and pose prediction with awareness of the underlying 3D shapes.

3. Overview

We study the problem of object reconstruction from a few RGB images without camera poses and category information. As Fig. 2 illustrates, our model FORGE learns camera pose estimation from 2D images and 3D features extracted from them. To deal with the large pose variation between the few views, our pose estimator jointly predicts the relative camera poses of all input images rather than chaining up pair-wise pose estimations. To handle objects from novel categories, we avoid learning category-specific priors. FORGE extracts per-view 3D features in their own camera spaces and transforms them into a shared reconstruction space with the estimated relative camera poses. Therefore we get rid of the category-specific canonical space for reconstruction. In the shared reconstruction space, we use a feature fusion module to aggregate information from single-view 3D features. Then a decoder head predicts a neural volume from the fused features. We use volume rendering techniques to produce the reconstruction results. During training, we render the results of input views and use 2D-based rendering loss as supervision. FORGE can be trained without any 3D supervision of object geometry. We design a new loss function for learning consistent 3D features across views. Besides, we can obtain accurate voxel reconstruction from the neural volume by a simple threshold. We introduce details of FORGE and the training protocol in the following.

4. Method

Here we introduce the design of FORGE. The input of the model is k image observations $\mathcal{I} = \{I^i | i = 1, ..., k\}$ captured by corresponding cameras $\mathcal{C} = \{C^i | i = 1, ..., k\}$. The model predicts: i) 3D features $\mathcal{Z}_{3D} = \{z^i_{3D} | i = 1, ..., k\}$ of all views; ii) relative camera poses $\Delta\Phi_i = \{\Delta\Phi^j_i | j = 1, ..., k; j \neq i\}$, $\Delta\Phi^j_1 \in \mathbb{SE}(3)$, where $\Delta\Phi^j_i$ is the pose of camera C^j relative to camera C^i, and the i^{th} frame is set as the canonical frame. Then the features are transformed by the relative camera poses into camera C^i's frame and fused to predict the neural radiance field V. We use the first frame as the canonical frame by default, if not otherwise specified. We elaborate on each component in the following.

4.1. Voxel Encoder

For a view I^i, the encoder F_{3D} encodes it into 3D voxel features $z_{3D} = F_{3D}(I)$, where $z^i_{3D} \in \mathbb{R}^{c \times d \times h \times w}$. In detail, we use ResNet-50 [10] to extract 2D feature maps $z^i_{2D} = F_{2D}(I)$, where $z^i_{2D} \in \mathbb{R}^{C \times h \times w}$. Then we reshape z^i_{2D} to 3D voxel features in $\mathbb{R}^{(C/d) \times d \times h \times w}$ as a deprojection operation. We finally perform one 3D convolution on it for refinement, which changes the 3D channel size to c.

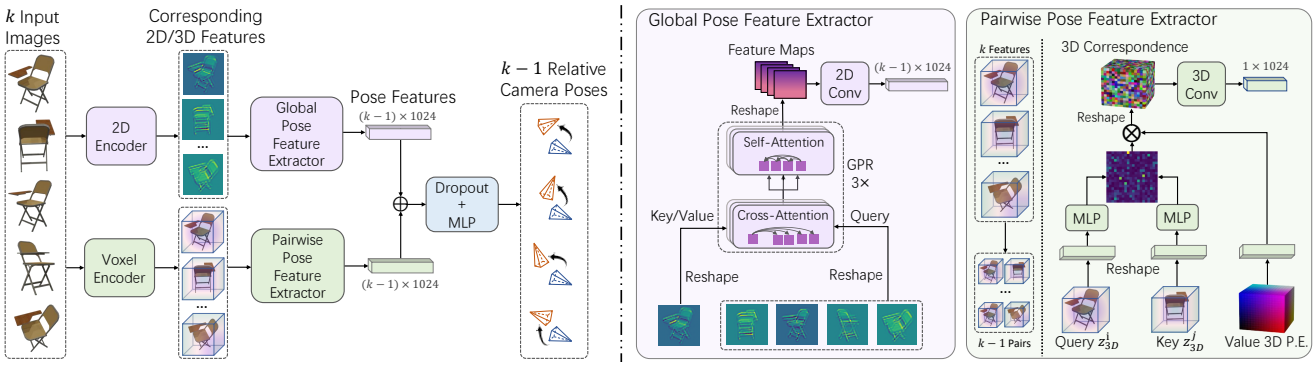


Figure 3. **Relative camera poses estimator.** FORGE uses a Global Pose Features Extractor to jointly reason about 2D correspondence cues for all frames. FORGE alternatively uses a Local Pose Features Extractor to predict pairwise pose features by explicitly reasoning about 3D correspondence.

4.2. Relative Camera Pose Estimator

Estimating relative camera poses can be an ill-posed problem under large pose variation — the shared visible object part between two views can be small, making it hard to establish cross-view correspondences. Moreover, building 2D correspondence on 2D images or feature maps [22, 42] is vulnerable under the 3D-2D projection ambiguity.

We introduce two novel pose feature extractors complementary to each other. As shown in Fig. 3, we use a global pose feature extractor, taking all 2D views as input and jointly reasoning about poses for all frames. We also build a pairwise pose feature extractor that computes explicit 3D correspondence on the predicted 3D features. The design benefits pose estimation in three ways: i) The global pose feature extractor allows information to pass between all frames. It leverages information from other views, which may have a larger shared visible part with the canonical view, to infer the relative pose of a query view. Besides, reasoning about poses globally in 2D is much cheaper than on 3D features; ii) Finding 3D correspondence on the predicted 3D features avoids the 3D-2D ambiguity problem and makes pose estimation more accurate; and iii) The multi-modal inputs (both 2D views and 3D features) of the pose estimator make it more robust.

Global Pose Feature Extractor. The global pose feature extractor takes in all views and jointly predicts pose features, denoted as $p_g \in \mathbb{R}^{(k-1)\times 1024}$, corresponding to all $k-1$ query views $\mathcal{I}_q$.

We use another 2D backbone to extract 2D feature maps of each view, denoted as $\mathbb{Z}'_{2D} = \{z'^i_{2D} | i = 1, ..., k\}, z'^i_{2D} \in \mathbb{R}^{h'\times w'\times 1024}$. We then reshape the canonical view features into a 1D vector $k_g \in \mathbb{R}^{N_{2D}\times 1024}$, where $N_{2D} = h' \cdot w'$. We similarly reshape the $k-1$ query view features as $q_g \in \mathbb{R}^{N_q\times 1024}$, where $N_q = (k-1)\cdot N_{2D}$. Then we use multiple global pose feature reasoning (GPR) modules to infer pose features. Specifically, each GPR module includes two standard multi-head transformer [52] blocks. In the first transformer block, we perform cross-attention, where the query is the feature q_g of the query view, and the key and value are features k_g of the canonical view. The block reasons 2D correspondence between each query view and the canonical view. Then we perform self-attention on the updated query view features. It jointly refines the correspondence cues for all query views by referring information from each other. After the GPR modules, we reshape the updated query view features back into 2D and use 2D convolutions to down-scale the 2D resolution to 1 to get the global pose features p_g.

Pairwise Pose Feature Extractor. The pairwise pose feature extractor builds $k-1$ feature pairs, where each pair consists of 3D features from a query view and the canonical view. Then it predicts pose features for them separately. Take the query view I^i and the canonical view I^1 as an example. The inputs are 3D voxel features z^i_{3D}, z^1_{3D}, and the output is the relative pose features $p^i_l \in \mathbb{R}^{1\times 1024}$.

Specifically, we reshape the 3D features into a 1-D vector in $\mathbb{R}^{N_{3D}\times c}$, where $N_{3D} = d \cdot h \cdot w$. Then we compute the similarity tensor $S^i = z^i_{3D} \cdot (z^1_{3D})^T$, where $(\cdot)^T$ is the transpose operation. We get the correspondence as $Corr^i_{1D} = S^i \cdot PE_{1D}$, where PE_{1D} is the expanded high-dimensional 3D positional embedding [52]. The positional embedding represents the location of each voxel in a high-dimensional space. $PE \in \mathbb{R}^{d\times h\times w\times c}$ and $Corr^i_{1D} \in \mathbb{R}^{N_{3D}\times c}$. We reshape the correspondence back into a 3D volume $Corr^i \in \mathbb{R}^{d\times h\times w\times c}$. For each voxel of the volume, its features represent the location of its corresponding voxel location within the canonical view 3D features in high-dimensional space. In the end, we use 3D convolutions to down-scale it to resolution 1 to get the relative pose features p^i_l of $\{I^i, I^1\}$. We concatenate the features of all query views as the final output $p_l \in \mathbb{R}^{(k-1)\times 1024}$.

Pose Prediction. We concatenate the pose features predicted by the two feature extractors to get the final features p. We use an MLP to regress the relative camera poses

$\Delta\Phi_i = \{\Delta\Phi_1^j | j = 2, ..., k\}$ for $k-1$ query views. To prevent the model from overfitting to the features extracted by either one, we use a Dropout layer before the MLP with a probability of 0.6 during training.

4.3. Feature Fusion Module

Feature Transformation. In general, we transform a 3D point in camera C^i's frame into camera C^j's frame by using rigid transformation $T_i^j = \Phi^j \cdot (\Phi^i)^{-1}$, where $(\cdot)^{-1}$ is the inverse operation and Φ is camera pose. Given the extracted 3D features of $k-1$ query views, we use the camera poses to transform them into the canonical view's camera frame. Specifically, the camera poses of the query views are computed by $\Phi^i = \Phi^1 \cdot \Delta\Phi_1^i$, where Φ_1 is a fixed canonical pose. We denote the 3D features after transformation as $\dot{\mathbb{Z}}_{3D} = \{\dot{z}_{3D}^i | i = 1, ..., k\}$.

View Fusion. For the transformed 3D features $\dot{\mathbb{Z}}_{3D}$, we first perform average pooling over them. The pooled 3D features serve as an initialization for sequentially fusing per-view features into final 3D features Z_{3D} using ConvGRU [1]. Specifically, the fusion sequence is determined by the relative camera poses, where we first fuse features of the views closer to the canonical view. When the predicted camera poses are noisy, the average pooling operation keeps the low-frequency signals, and the sequential fusion recovers high-frequency details.

4.4. Neural Volume-based Rendering

Inspired by NeRF [26], we reconstruct objects with radiance fields. These fields, represented as neural volumes, are predicted in a fully *feed-forward* manner.

Definition and Prediction. We denote the neural volume as $V := (V_\sigma, V_f)$, where V_σ is the density values, and V_f is the neural features. They help reconstruct the geometry and appearance of the object. Given the 3D feature volume Z_{3D} after fusion, we use two prediction heads composed of several 3D convolution layers to predict the V_σ and V_f, respectively.

Volume Rendering. We read out the predicted neural volume on 2D using volume rendering as $(\hat{I}, \hat{I}_\sigma) = \pi(V, \Phi)$, where Φ is the camera pose, $\hat{I}$ and $\hat{I}_\sigma$ are rendered image and mask. We follow the volume rendering techniques in NeRF [26]. Differently, for each 3D query point p, we get its 3D features by interpolating the neighbor voxel grids. Besides, we first render a feature map and then use several 2D convolutions to predict the final RGB image [29].

4.5. Training Protocol

We train FORGE in three stages. First, we train the voxel encoder (Sec. 4.1), fusion module (Sec. 4.3), and volume renderer (Sec. 4.4) under ground-truth poses using $L_{3D} = L_{mv} + L_{corr}$. Specifically, L_{mv} is the 2D photometric loss applied on all input views.

$$L_{2D} = ||I_\sigma^i - \hat{I}_\sigma^i|| + \lambda_{img}||I^i - \hat{I}^i||, \quad (1)$$

$$L_{mv} = \frac{1}{k}\Sigma_{i=1}^k(L_{2D}(I_\sigma^i, \hat{I}_\sigma^i, I^i, \hat{I}^i) + \lambda_p L_p(I^i, \hat{I}^i)), \quad (2)$$

The cross-view consistency loss L_{corr} is designed to encourage the features that correspond to the same object part while extracted from a different view to be close in the feature space, which makes the reconstruction result more coherent. The well-established feature space also benefits following correspondence-based pose estimation, as finding cross-view correspondence using our similarity-based method becomes more effortless in the feature space. We compute the loss on rendering results. We separate the input views into two sets: we use one set (n views) for building the neural volume and use the cameras of the other set ($k-n$ views) to render the results and vice versa. The reconstruction of views in the second set should be reasonable.

$$L_{corr} = \frac{1}{k-n}\Sigma_{i=n+1}^k L_{2D}(I_\sigma^i, \ddot{I}_\sigma^i, I^i, \ddot{I}^i), \quad (3)$$

where $\ddot{I}$ and $\ddot{I}_\sigma$ are results rendered using a subset of views as input.

Second, we train the relative camera pose estimator with a loss $L_{pose} = ||\Phi - \hat{\Phi}||^2$, where Φ, $\hat{\Phi}$ are ground-truth and predicted relative camera poses. The two pose feature extractors are trained separately and then fine-tuned together.

Finally, we fine-tune the model end-to-end with all losses above. We empirically observe that single-stage training leads to collapse. We conjecture that the pose estimator relies on representations from a well-initialized voxel encoder.

5. Experiments

As previous datasets [45, 57] contain a limited number of object categories or the images are captured under restricted camera settings, we design a new dataset to evaluate the performance of FORGE.

Kubric Synthetic Dataset. We generate a new synthetic dataset using Kubric [9], which contains 23 categories from ShapeNet [5]. For the training set, we randomly select 1000 objects instances from each of the 13 categories, having 13,000 instances in total. We *randomly* synthesize camera poses to render observations. The randomly sampled cameras make the dataset more realistic and challenging than the previous one that uses fixed and evenly distributed camera poses [45]. We build **two test sets**. The first one contains 100 instances for each of 13 *training categories* (1300 in total), where 50 instances are *seen* during training and 50 are *unseen*. The second one contains 100 instances for each of 10 *novel categories* (1000 in total). We use 5 views for input and another 5 for evaluation.

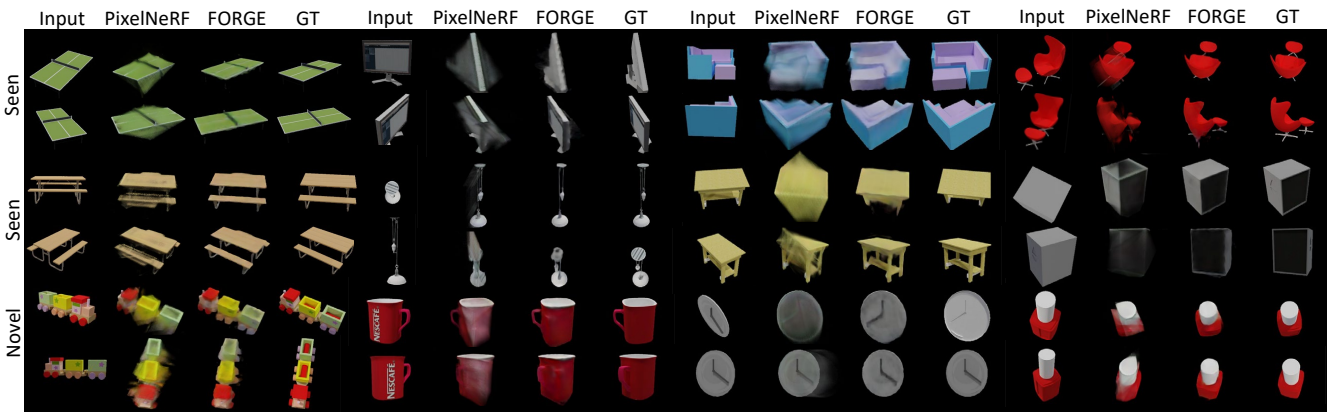


Figure 4. **Reconstruction results under ground-truth poses.** We show examples of both seen training categories and novel test categories. For each instance, we show two of the five inputs and two of the five novel views due to space limit. See supplementary for the full results.

			PixelNeRF [63]		SRT [40]		FORGE		FORGE†	
Dataset	*Cat.*	*Ins.*	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Test1-seen	*13*	*650*	29.22	0.893	21.81	0.822	**31.28**	**0.938**	**26.59**	**0.887**
Test1-unseen	*13*	*650*	29.27	0.894	21.86	0.820	**31.36**	**0.938**	**26.72**	**0.887**
Test2-novel	*10*	*1000*	29.37	0.906	19.25	0.817	**31.17**	**0.946**	**25.85**	**0.891**

Table 2. Evaluation of reconstruction quality on Kubric synthetic dataset. † means using predicted poses, test-time optimization and canonical frame selection. Models without † use G.T. relative camera poses.

	Time
P-NeRF [63]	3.1 min
SRT [40]	0.35 sec
FORGE	**0.06 sec**
FORGE*	**0.09 sec**

Table 3. Inference time comparison.

Implementation Details. The resolution of inputs is 256×256. We use $k = 5$ images, separated into 2 and 3 views, to compute the cross-view consistency loss. We set $\lambda_{img} = 5$, $\lambda_p = 0.02$ and $\lambda_{pose} = 1$. The neural volume contains 64^3 voxels. We sample 64 points on each ray for rendering. Please refer to the supplementary for more training details.

Metrics. We adopt standard novel view synthesis metrics PSNR (in dB) and SSIM [56] to evaluate reconstruction results. We also evaluate relative camera pose error.

Our Tested Models

- FORGE. Our model uses ground-truth relative poses with the first view as the canonical view.
- FORGE*. Our model uses predicted relative poses with the first view as the canonical view.
- FORGE†. Our model with canonical view selection and test-time optimization based on FORGE*. We optimize the predicted pose using 2D photometric loss L_{2D} (Sec. 4.5) on the input views. We perform k times inference by setting every frame as canonical and use L_{2D} as the selection criterion.

Baselines. For reconstruction, we compare with PixelNeRF [63] and SRT [40] under ground-truth poses to validate the ability of FORGE on learning 3D geometry priors. For relative pose estimation, we compare with i) 2D-based VideoAE [17], 8-Point TF [39], which separately predicts poses for each query view; ii) 2D-based RelPose [64] which jointly estimate all relative poses, and iii) 3D-based Gen6D which separately predicts poses for each query view. All pose estimation baselines are trained with ground-truth poses.

5.1. Reconstruction Evaluation

Qualitative Results under Ground-Truth Pose. As shown in Fig. 4, we compare FORGE with PixelNeRF [63] on both seen categories during training and novel testing categories. We observe that results of PixelNeRF [63] have strong artifacts when the input and target views have dramatic pose changes. PixelNeRF [63] is also vulnerable under occlusion, while results of FORGE are more consistent. We conjecture that PixelNeRF [63] is built on 2D representations, which cannot handle 2D ambiguity well. In comparison, FORGE benefits from our 3D-aware designs.

Qualitative Results under Predicted Pose. We show the results in Fig. 5. FORGE can reconstruct the objects under noisy predicted poses. The geometry reconstruction performance on delicate objects and novel object categories match the results using ground truth poses. Besides, we visualize the voxel reconstruction based on the predicted density volume in Fig. 6. It reveals that FORGE can capture 3D geometry accurately.

Real-life Object Reconstruction. We demonstrate that FORGE can reconstruct real-life objects from images in Fig. 7. We use three views as input and an off-the-shelf method [14] to get object masks. Even though the domain gap exists, such as different camera intrinsics, FORGE can produce reasonable results.

Comparison with State-of-the-Art. As shown in Table 2, when using ground truth poses, FORGE outperforms Pix-

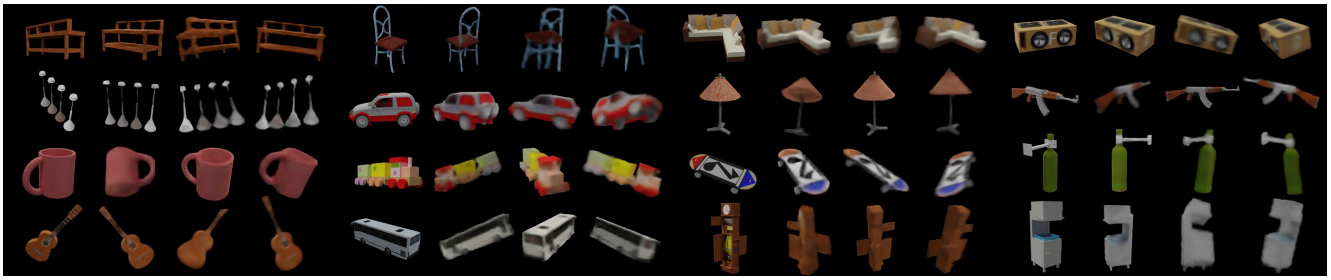


Figure 5. **Reconstruction results under predicted poses.** For each example, we show four images, where the first is one of the inputs, and the last three are reconstructions. The first and last rows are examples from seen and novel categories, respectively.

Figure 6. **Visualization of voxel reconstruction under predicted poses.** We obtain the voxel reconstruction by thresholding the predicted densities. Note that the thresholds are varied for different examples. We show the voxels in two views for each example.

	VideoAE [17]		8-Point TF [39]		RelPose [64]		Gen6D Refiner [21]		FORGE*		FORGE†	
Dataset	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓
Test1-seen	36.7	0.56	15.0	0.32	24.7	-	17.2	0.36	**7.3**	**0.17**	**4.9**	**0.12**
Test1-unseen	33.8	0.57	15.4	0.32	24.0	-	17.3	0.35	**7.9**	**0.18**	**5.6**	**0.14**
Test2-novel	42.7	0.69	23.9	0.49	32.3	-	22.8	0.52	**13.7**	**0.31**	**10.6**	**0.26**

Table 4. Evaluation of relative camera pose estimation on Kubric synthetic dataset. All models are trained with ground-truth poses.

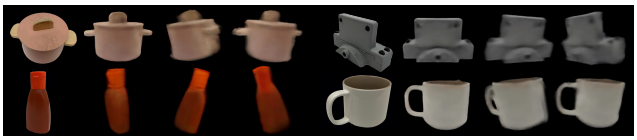


Figure 7. **Reconstruction results on real-world images.** We use three images as input. We show one input view and reconstruction in three novel views. All objects are from novel categories.

elNeRF [63] and SRT [40] by a significant margin on both seen and novel object categories. Furthermore, the inference speed of FORGE is much faster than PixelNeRF, as shown in Table 3. When using predicted poses, FORGE† matches the performance using ground truth poses. The performance gap between seen and novel object categories is small, showing the strong generalization ability of FORGE.

5.2. Pose Estimation Evaluation

Qualitative Results. We visualize pose estimation results in Fig. 8. Our prediction matches ground truth accurately.

Comparison with State-of-the-Art. As Table 4 shows, FORGE* achieves remarkable gain over previous methods. Both rotation and translation error is half of the best previous method 8-Point Transformer [39]. After test-time optimization, FORGE† achieves 30% performance gain. Moreover, our method shows a much smaller generalization gap, e.g., rotation error gap 5 degree of FORGE† vs. 9 degree of [39] on seen and novel object categories.

5.3. Ablation Study

In this section, we verify the effectiveness of each proposed module on the Kubric synthetic dataset.

Voxel Encoder Design. We validate our design of the voxel encoder by comparing it with a counterpart using an additional 3D U-Net for feature refinement. As shown in Table 5, using more 3D convolutions improves performance on seen object categories. However, the performance drops dramatically on novel object categories. This phenomenon implies that 3D U-Net overfits the training categories, which weakens its generalization ability. Instead, our encoder design shows great power in learning generalizable 3D priors.

Cross-view Consistency Loss for Learning 3D Representation. We study the impact of the cross-view consistency loss (CCL) to train the voxel encoder. As shown in Table 5, the PSNR and SSIM drop 2.5 dB and 0.25, respectively, without using the cross-view consistency loss. As visualization shown in Fig. 9a, using CCL improves the reconstruction quality on object parts that are visible but heavily occluded or only visible in one of the input views. It demonstrates that CCL regularizes the learning of 3D features by cross-view rendering.

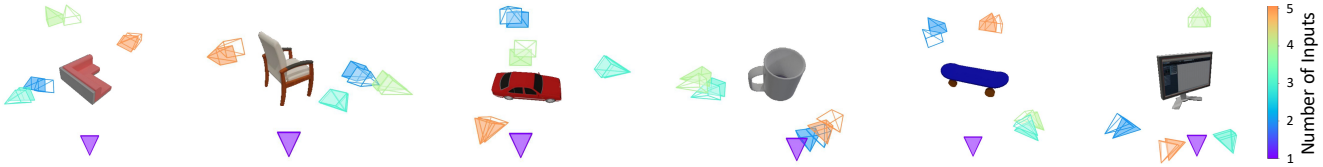


Figure 8. **Visualization of pose estimation.** Prediction and ground-truth are shown in the same colors with colored and white faces.

	w/ 3D U-Net [68]		w/o CCL		FORGE	
Dataset	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Test1	**32.75**	**0.950**	28.78	0.914	31.32	0.938
Test2	22.33	0.826	28.05	0.918	**31.17**	**0.946**

Table 5. Ablation on voxel encoder design and the use of cross-view consistency loss (CCL). Experiments use G.T. poses.

	Global		Pairwise		Global-Pairwise	
Dataset	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓
Test1	12.9	0.24	11.3	0.21	**8.0**	**0.17**
Test2	26.2	0.44	22.1	0.42	**13.8**	**0.31**

Table 6. Ablation on the two feature extractors of pose estimation module. The models are not jointly trained with the voxel encoder.

	Global w/o JR		Pairwise w/o CCL		Pairwise-implicit	
Dataset	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓	Rot. ↓	Trans. ↓
Test1	15.1	0.28	20.1	0.33	23.4	0.37
Test2	30.2	0.52	34.7	0.60	38.5	0.66

Table 7. Ablation on the designs of two pose feature extractors.

	only Avg. Fusion		only Seq. Fusion		both (FORGE*)	
Dataset	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Test1	24.23	0.852	25.36	0.864	26.08	0.879
Test2	23.48	0.849	24.80	0.860	25.11	0.880

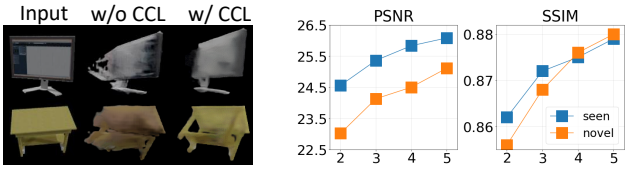
Table 8. Ablation on fusion methods and test-time optimization.

Pose Estimator Design. As Table 6 shows, using either the global or pairwise pose feature extractor to predict poses achieves better performance than previous works in Table 4. Using both contribute to a stronger relative pose estimator.

Joint Pose Reasoning. We validate the effectiveness of performing joint pose reasoning (JR) using the self-attention blocks for the global pose feature extractor. For each global pose reasoning (GPR) module in the global pose feature extractor, we remove the self-attention block and double the cross-attention block, thus retaining the same model capacity. As shown in Table 7, without joint pose reasoning, the performance of the global pose estimator drops 15% compared to Global in Table 6.

Cross-view Consistency Loss for Accurate Correspondence. For the pairwise pose features extractor, we first study the importance of using CCL to build 3D correspondence. As shown in Table 7, when using the encoder trained without CCL, the performance drops 60% compared to Pairwise in Table 6. It reveals the importance of enforcing cross-view consistency for finding reliable correspondence.

Explicit 3D Correspondence. We also verify that leveraging explicit 3D correspondence is helpful for relative pose estimation. We compare the pairwise pose feature extractor with a counterpart that implicitly finds the correspondence, denoted as Pairwise-implicit. For Pairwise-implicit, the value of attention is the canonical view features rather than the 3D positional encoding. As shown in Table 7, the performance drops dramatically compared to the original model Pairwise in Table 6. We conjecture that finding explicit correspondence helps mapping 3D voxel features to another feature space composed by the positional embedding. The feature space is consistent across different inputs, making the learning of pose regression easier than only using voxel features.



(a) (b)

Figure 9. (a) FORGE reconstructs rarely visible object parts better with cross-view consistency loss (CCL); (b) Ablation on the number of input views on both seen and novel categories.

Fusion Module Design. We validate our designs on the feature fusion module by comparing the full fusion model with counterparts using (i) only average pooling fusion, and (ii) only sequential fusion. As shown in Table 8, using average view-pooling fused features as initialization improves the sequential fusion.

Number of Inputs. As shown in Fig. 9b, we test the robustness of our model in handling varying numbers of input views from two to five. The performance is improved with more input views, and it tends to saturate with five.

6. Conclusion

We study the problem of reconstructing objects from a few views, where both the object categories information and camera poses are unknown. Our key insight is leveraging the synergies between shape reconstruction and pose estimation to improve the performance of both tasks. We design a voxel encoder and a relative camera pose estimator, which are trained in a cascade manner. The pose estimator jointly reasons the poses of all views and establishes explicit 3D cross-view correspondences. Then the 3D voxel features extracted from each view are transformed into a shared reconstruction space using the predicted poses. Then a neural volume is predicted, fusing per-view information. Our model

outperforms prior art on reconstruction and relative pose estimation by a significant margin. Ablation studies also show the effectiveness of our designs for each module. We hope that our work inspires future efforts in making object reconstruction broadly applicable and scalable for capturing daily objects in the real world.

References

[1] Nicolas Ballas, L. Yao, Christopher Joseph Pal, and Aaron C. Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[2] Mohamed El Banani, Luya Gao, and Justin Johnson. Unsupervisedr&r: Unsupervised point cloud registration via differentiable rendering. In *CVPR*, pages 7125–7135, 2021.

[3] Mafkereseb Kassahun Bekele, Roberto Pierdicca, Emanuele Frontoni, Eva Savina Malinverni, and James Gain. A survey of augmented, virtual, and mixed reality for cultural heritage. *Journal on Computing and Cultural Heritage (JOCCH)*, 11(2):1–36, 2018.

[4] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. In *CVPR*, pages 2560–2568, 2018.

[5] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015.

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14104–14113, 2021.

[7] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016.

[8] Rohit Girdhar, David F. Fouhey, Mikel D. Rodriguez, and Abhinav Kumar Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

[9] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. *ArXiv*, abs/2203.03570, 2022.

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016.

[11] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *arXiv preprint arXiv:2104.01542*, 2021.

[12] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[13] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NIPS*, 2017.

[14] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross B. Girshick. Pointrend: Image segmentation as rendering. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9796–9805, 2020.

[15] Georg S. W. Klein and David William Murray. Parallel tracking and mapping on a camera phone. *IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86, 2009.

[16] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Kumar Gupta. 3d-relnet: Joint object and relational network for 3d prediction. In *ICCV*, pages 2212–2221, 2019.

[17] Zihang Lai, Sifei Liu, Alexei A. Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. In *ICCV*, pages 9710–9720, 2021.

[18] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, pages 5721–5731, 2021.

[19] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[20] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020.

[21] Yuan Liu, Yilin Wen, Sida Peng, Chu-Hsing Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. *ArXiv*, abs/2204.10776, 2022.

[22] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[23] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[24] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, pages 7206–7215, 2021.

[25] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4455–4465, 2019.

[26] 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. In *ECCV*, 2020.

[27] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. Openmvg: Open multiple view geometry. In *RRPR@ICPR*, 2016.

[28] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, pages 7587–7596, 2019.

[29] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11448–11459, 2021.

[30] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, pages 3501–3512, 2020.

[31] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, pages 5569–5579, 2021.

[32] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019.

[33] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *CVPR*, pages 10707–10716, 2020.

[34] Keunhong Park, U. Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5845–5854, 2021.

[35] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10313–10322, 2021.

[36] Hafizur Rahaman, Erik Champion, and Mafkereseb Bekele. From photo to 3d to mixed reality: A complete workflow for cultural heritage visualisation and experience. *Digital Applications in Archaeology and Cultural Heritage*, 13:e00102, 2019.

[37] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotný. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, pages 10881–10891, 2021.

[38] Gernot Riegler, Ali O. Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, pages 6620–6629, 2017.

[39] Chris Rockwell, Justin Johnson, and David F. Fouhey. The 8-point algorithm as an inductive bias for relative pose prediction by vits. In *3DV*, 2022.

[40] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. *CVPR*, 2022.

[41] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016.

[42] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[43] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, volume abs/2007.02442, 2020.

[44] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 1:519–528, 2006.

[45] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019.

[46] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[47] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. In *ICLR*, 2019.

[48] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. In *ICLR*, 2020.

[49] Shubham Tulsiani, Saurabh Gupta, David F. Fouhey, Alexei A. Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, pages 302–310, 2018.

[50] Shubham Tulsiani, Tinghui Zhou, Alyosha A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[51] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *CVPR*, pages 2590–2598, 2019.

[52] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[53] He Wang, Srinath Sridhar, Jingwei Huang, Julien P. C. Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2637–2646, 2019.

[54] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[55] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, pages 4688–4697, 2021.

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

[57] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82, 2014.

[58] Haozhe Xie, Hongxun Yao, Shengping Zhang, Shangchen Zhou, and Wenxiu Sun. Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images. *International Journal of Computer Vision*, pages 1 – 17, 2020.

[59] Xinchen Yan, Jasmined Hsu, Mohammad Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. Learning 6-dof grasping interaction via deep geometry-aware 3d representations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3766–3773. IEEE, 2018.

[60] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016.

[61] Zhenpei Yang, Zhile Ren, Miguel Angel Bautista, Zaiwei Zhang, Qi Shan, and Qixing Huang. Fvor: Robust joint shape and pose optimization for few-view object reconstruction. In *CVPR*, 2022.

[62] Yufei Ye, Shubham Tulsiani, and Abhinav Kumar Gupta. Shelf-supervised mesh prediction in the wild. In *CVPR*, pages 8839–8848, 2021.

[63] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4576–4585, 2021.

[64] Jason Y. Zhang, Deva Ramanan, and Shubham Tulsiani. RelPose: Predicting probabilistic relative rotation for single objects in the wild. In *European Conference on Computer Vision*, 2022.

[65] Jason Y. Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. In *NeurIPS*, 2021.

[66] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020.

[67] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *CVPR*, 2022.

[68] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S...*, 11045:3–11, 2018.

[69] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, 2022.

Appendices

A. Implementation Details

In this section, we introduce the details of our model FORGE, including the voxel encoder, relative camera pose estimator, across-view feature fusion module, and volume rendering techniques.

A.1. Voxel Encoder

The input of the 3D encoder is the RGB image observation and the output is per-view 3D features. The architecture is shown in Table 9. For each 3D convolution layer, we use batch normalization with leaky ReLU activation.

Stage	Configuration	Output
0	Input image	$256 \times 256 \times 3$
	2D Feature extraction	
1	Res-50 (stride 8) [10]	$32 \times 32 \times 2048$
	2D to 3D Convertion	
2	Reshape	$32 \times 32 \times 32 \times 64$
	3D Feature Processing	
3	Conv3D	$32 \times 32 \times 32 \times 128$

Table 9. Network architecture of the voxel encoder.

A.2. Relative Camera Poses Estimator

The relative camera poses estimator is composed by two feature extractor and a pose regression head. Their architecture are shown below.

Global Pose Feature Extractor. The Global Pose Feature Extractor infers on 2D features of all inputs, which jointly reasons pose features of all views. The architecture is shown in Table 10. In detail, the positional embedding added to each query view features are the same.

Pairwise Pose Features Extractor. The Pairwise Pose Features Extractor infers on 3D features of inputs in a pairwise manner. The architecture is shown in Table 11.

Pose Regression. The architecture of the pose regression head is shown in Table 12.

A.3. Feature Fusion Module

The feature fusion module is composed by an average fusion block and a sequential fusion block. The inputs of the feature fusion module are the 3D features that have been transformed into the same reconstruction space, denoted as $\dot{\mathbb{Z}}_{3D} = \{\dot{z}^i_{3D} | i = 1, ..., k\}$.

Stage	Configuration	Output
0	Input image (N views)	$k \times 256 \times 256 \times 3$
	2D Feature extraction	
1	FPN-P4 [19]	$k \times 16 \times 16 \times 256$
	Get Features	
2	Query view	$(k-1) \times 16 \times 16 \times 256$
2	Canonical features	$1 \times 16 \times 16 \times 256$
	Reshape	
3	Query views	$[(k-1) \cdot 16 \cdot 16] \times 256$
3	Canonical view	$[16 \cdot 16] \times 256$
	Add Pos. Embedding	
4	Query views	$[(k-1) \cdot 16 \cdot 16] \times 256$
4	Canonical view	$[16 \cdot 16] \times 256$
	Global Pose Reasoning ($\times 3$)	
5	Cross-attention	$[(k-1) \cdot 16 \cdot 16] \times 256$
6	Self-attention	$[(k-1) \cdot 16 \cdot 16] \times 256$
	Reshape	
7	Query views	$(k-1) \times 16 \times 16 \times 256$
	Get Pose Features	
8	2D Convs	$(k-1) \times 1 \times 1 \times 1024$
9	Reshape	$(k-1) \times 1024$

Table 10. Network architecture of the Global Pose Features Extractor.

Stage	Configuration	Output
0	Query view features	$32 \times 32 \times 32 \times 128$
0	Canonical view features	$32 \times 32 \times 32 \times 128$
0	3D Positional Encoding	$16 \times 16 \times 16 \times 64$
	3D Feature Volume Processing	
1	2 Conv3D Layers	$16 \times 16 \times 16 \times 64$
	Finding 3D correspondence	
2	Cross-view Transformer with P.E.	$16 \times 16 \times 16 \times 64$
	Correspondence Refinement	
3	Self-Attention	$16 \times 16 \times 16 \times 64$
	Get pose features	
4	Conv3D Layers	$1 \times 1 \times 1 \times 1024$
5	Reshape	1×1024

Table 11. Network architecture and of the Pairwise Pose Features Extractor.

Stage	Configuration	Output
0	Features (Global)	$(k-1) \times 1024$
0	Features (Pairwise)	$(k-1) \times 1024$
	Merge Features	
1	Concatenation	$(k-1) \times 2048$
	Pose Regression	
2	Dropout + MLP	$(k-1) \times 7$

Table 12. Network architecture and of the pose regression head.

First, we perform average pooling over the features as $Z_{3D} = Avg(\dot{\mathbb{Z}}_{3D})$, where $Avg(\cdot)$ is the average pooling

operation for all views.

Then, we fuse the per-view 3D features to Z_{3D} sequentially using ConvGRU. After we permute the sequence of 3D features by the distance between canonical view and the correspdoning view, the sequential fusion can be denoted as $Z_{3D} = GRU(Z_{3D}, \dot{\mathbb{Z}}_{3D})$, where GRU is the ConvGRU, and Z_{3D} serves as a feature initialization. Please refer to [1] for more details of sequential fusion using ConvGRU.

A.4. Volume Rendering

We perform parameterized volume rendering as render 3D features on 2D, and then perform 2D convolutions to get the final 2D reconstruction results. The input of the volume rendering is the 3D feature after cross-view fusion. We predict the neural volume $V := (V_\sigma, V_f)$, where V_σ, V_f are density volume and feature volume, from the 3D feature and use it for rendering. The architecture is shown in Table 13.

Stage	Configuration	Output
0	3D Feature	$32 \times 32 \times 32 \times 128$
	Predict Neural Volume	
1	Density Volume	$64 \times 64 \times 64 \times 1$
1	Feature Volume	$64 \times 64 \times 64 \times 16$
	Feature Volume Rendering	
1	Volume Renderer	$128 \times 128 \times 16$
	Feature Map to 2D Results	
2	Conv2D Layers	$256 \times 256 \times 4$
	Get RGB and mask	
3	RGB	$256 \times 256 \times 3$
3	Mask	$256 \times 256 \times 1$

Table 13. Network architecture and of the volume renderer.

B. Dataset

We introduce more details of the Kubric Synthetic Dataset. To generate 2D observations, we randomly sample the cameras rotations. The camera are assigned with a distance to the object within a range of $[1.4, 1.6]$. we change the size of the objects by normalizing their axis-aligned bounding box to 1^3 meter.

Training Categories. We include 13 training categories, they are: *airplane, bench, cabinet, car, chair, display, lamp, loudspeaker, rifle, sofa, table, telephone and vessel*. All of these 13 categories have more than 1000 instances, ensuring the diversity of the datasets. We randomly sample 1000 instances of each category for training.

Unseen Testing Categories. We include 10 novel testing categories not seen during training, they are: *bus, guitar, clock, bottle, train, mug, washer, skateboard, dishwasher, and pistol*. We randomly sample 100 instances of each category for testing the generalization ability of FORGE.

C. Training Details

In this section, we introduce more details on training FORGE. We first train the model except the relative camera pose estimator. We use batch size 32 with learning rate 0.0008 for $40,000$ iterations. We half the learning rate at $15,000$, $30,000$ iterations. Then we train the relative camera pose estimator with fixing other parts of the model. We use batch size 40 with learning rate 0.0002 for $130,000$ iterations, where the learning rate is half at $50,000$, $70,000$, $90,000$ and $110,000$ iterations. Specifically, we train the Global Pose Features Extractor and Pairwise Pose Features Extractor separately and then finetune them together. In the end, we train the model with batch size 32 and learning rate 0.0002 for $30,000$ iterations, where the learning rate is half at $10,000$ and $20,000$ iterations. For validation, we use a small dataset with 50 instances of training categories. We train our models with 8 A40 GPUs.

For volume rendering, The size of the predicted 3D volume is set to 1^3 meter. For each camera frame, the volume is located at the center of the camera's frame, and the camera itself has a pose $[\mathbf{R}|\mathbf{t}]$. $\mathbf{R}$ is an identity matrix and $\mathbf{t}$ is set to $[0, 0, 1.5]$.

D. More Results

We show the comparison with [63] in Fig. 10 with all five inputs and five novel view reconstruction results.

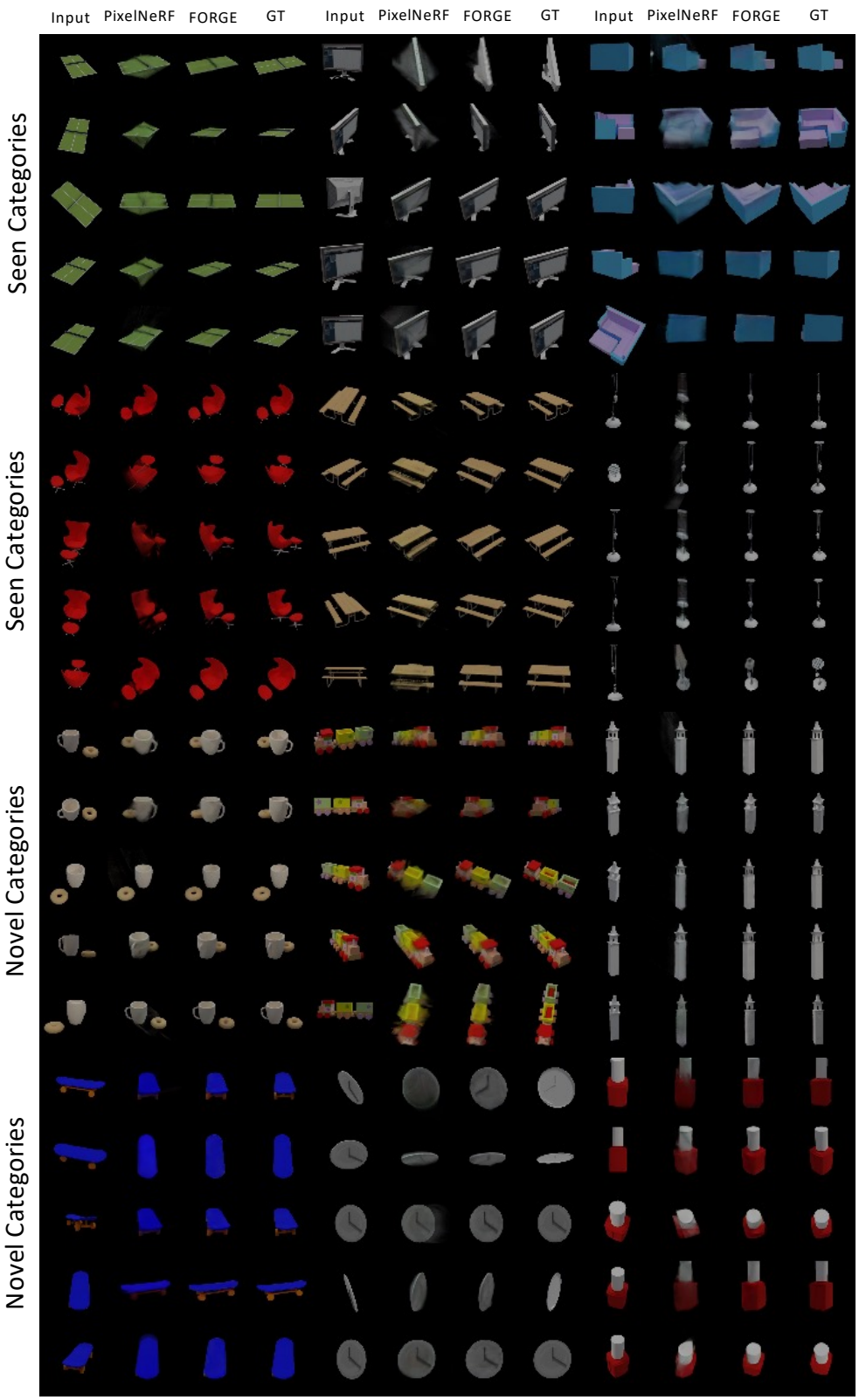


Figure 10. Comparison with PixelNeRF [63] using ground-truth camera poses. We show five inputs and five reconstruction views.