Common Pets in 3D:
Dynamic New-View Synthesis of Real-Life Deformable Categories

Sarth Samih
University of Toronto
samarth.sinha@mail.utoronto.ca

Roman Shapoalov
Meta AI
romansh@meta.com

Jeremy Reizenstein
Meta AI
reizenstein@meta.com

Ignacio Rocco
Meta AI
irocco@meta.com

Natalia Neverova
Meta AI
nneverova@meta.com

Andrea Vedaldi
Meta AI
vedaldi@meta.com

David Novotny
Meta AI
dnovotny@meta.com

Abstract

Obtaining photorealistic reconstructions of objects from sparse views is inherently ambiguous and can only be achieved by learning suitable reconstruction priors. Earlier works on sparse rigid object reconstruction successfully learned such priors from large datasets such as CO3D. In this paper, we extend this approach to dynamic objects. We use cats and dogs as a representative example and introduce Common Pets in 3D (CoP3D), a collection of crowd-sourced videos of approximately 4,200 distinct pets. CoP3D is one of the first large-scale datasets for benchmarking non-rigid 3D reconstruction “in the wild”. We also propose Tracker-NeRF, a method for learning 4D reconstruction from our dataset. At test time, given a small number of video frames of an unseen object, Tracker-NeRF predicts the trajectories of its 3D points and generates new views, interpolating viewpoint and time. Results on CoP3D reveal significantly better non-rigid new-view synthesis performance than existing baselines. The data will be available on the project webpage: https://cop3d.github.io/.

1. Introduction

Advances in photorealistic reconstruction and new-view synthesis facilitate experiencing real-life objects and scenes in virtual and mixed reality. However, compared to standard 2D photography, capturing 3D content remains significantly more difficult. Methods based on neural radiance fields [22, 24, 27], signed distance functions [57], and other implicit representations [43] use deep neural networks to represent the geometry of a single scene [27]. They usually require hundreds of input views for high-quality reconstruction, and are often limited to rigid objects and static scenes. Instead, we would like users be able to capture 3D and 4D content as easily as taking an image or video with their smartphone, a setting that we call casual capture.

Casual capture requires reconstructing objects from only a few images, which is inherently ambiguous. This is particularly true for dynamic content which requires to reconstruct not only shape and appearance, but also deformation. The ambiguity induced by deformations can be controlled via regularization [33, 47], but this can be too weak or too restrictive to work well in all cases. Alternatively, one can
adopt parametric models of motion such as linear blend skin-
ing [13,23,36,44,51,52], but these are difficult to generalise
beyond a few object categories such as humans. Learning-
based methods [2,9,11,29,38,58] address the reconstruction
ambiguity by learning 3D priors for specific types of objects
using large datasets such as CO3D [39]. With these priors,
they can achieve plausible 3D reconstructions from a small
number of images, but so far only for rigid objects.

In this paper, we wish to further extend data-driven ap-
proaches to dynamic objects which change their shape during
capture and thus require a 4D reconstruction. Our hypothe-
sis is that dynamic objects can be reconstructed even from a
monocular video, provided that one leverages priors learnt
from a large collection of relevant videos.

In order to test this hypothesis, we first contribute a
new crowd-sourced dataset, Common Pets in 3D (CoP3D).
CoP3D contains 4,200 videos of different cats and dogs. It
contains more than 600,000 video frames with 3D camera
tracking obtained using Structure-from-Motion (SfM) and
foreground object masks for each frame. The videos are
captured in the wild by non-experts, using smartphone cam-
eras, and are thus representative of the casual capture setting.
Like CO3D, CoP3D can be used for single-sequence and
category-wise reconstruction.

Our second contribution is Tracker-NeRF (TrackeRF),
a new method for few-view reconstruction of dynamic ob-
jects. The method is inspired by approaches such as Warped
Ray Embedding [10], PixelNeRF [58], and NeRFormer [38]
in that it learns to synthesise new views by triangulating
features extracted from the provided input views. Our key
innovation is modelling the deformations of objects by infer-
ing the 3D trajectory of each queried 3D point. Empirically,
we show that TrackeRF outperforms previous approaches
for new-view synthesis of dynamic objects.

2. Related work

Dynamic new-view synthesis. Various scene representa-
tions have been explored for new-view synthesis, in-
cluding meshes [19, 49], voxels [6, 15, 25, 40], radiance
fields [27, 30, 45], and sign-distance functions [32, 35, 43].
Most of these works have considered static scenes, but a few
have explored extensions to dynamic scenes.

The simplest extension is to add time as an additional
parameter of the representation [21, 54]. For example,
NSPF [21] adds time to NeRF, defining a space-time ra-
diance field. The latter is regularized by fitting a scene flow
field, using it to enforce time consistency. Video-NeRF [54]
fits a space-time radiance field too, but regularizes it by using
a monocular depth predictor.

An alternative approach is to use a deformation field
in order to reconstruct the different video frames
to a canonical representation which (approximately) time-
variant [5, 24, 37, 48]. Neural Volumes [24] learns an auto-
coder model mapping several input frames to an instanta-
aneous low-dimensional latent code; the latter is decoded into
a radiance field which is approximately time invariant and
a corresponding deformation field, expressed as a mixture
of a small number of local affine transformations. Closer to
NeRF, D-NeRF [37] does not use an auto-encoder, but fits
a canonical neural radiance field directly to a single video to-
gether with a corresponding deformation field. Nerfies [33]
fits an auto-decoder instead, learning small codes parameter-
ing the appearance and deformation of the individual
video frames. The deformation is given by a field of SE(3)
transformations controlled by an elastic regularization term.
Non-rigid NeRF [47] also uses an auto-decoder, but only to
express deformations, which controls the magnitude, and
discourages stretching and squeezing.

Other methods regularize the scene flow by exploiting
pre-trained optical flow predictors such as RAFT [46]. For
instance, LASR [55] fits a mesh predictor to a single video of
a deformable object and supervises it by using a combination
of photometric and optical flow losses.

Category-specific new-view synthesis. Like NeRF, most
of the methods above start from a ‘tabula rasa’, fitting a model
to a single video sequence. However, many recon-
struktion tasks are inherently ambiguous and can be
approached successfully only by combining evidence with
prior information. One way to do so is to learn an auto-
encoder from many videos of a given object category. Most
such methods work by fitting a parametric model of an ar-
ticulated object. CMR [14] and [17] rely on sparse keypoint
supervision to deform a template mesh to fit a given image.
UMR [20] does not require keypoints but assumes part seg-
mentation learned in a self-supervised manner. DOVE [53]
trains an auto-encoder from many videos of an object cate-
gory. The decoder is fixed and given by a deformable texture
mesh model of the object. Optical flow is used as additional
supervision. BANMo [56] fits a volumetric model to mul-
tiple videos of a single object instance, warping it using neural
blend skinning. This method also uses category-level priors
in the form of pre-trained canonical surface embeddings [31].

Sparse new-view synthesis. The methods above require
many input images for each reconstructed object. Recon-
struction from few views, or even a single view, further emphases the importance of priors. PixelNeRF [59] and
WCE [10] learn a category-specific multi-view auto-encoder
for a radiance field. In order to predict colour and occu-
pancy of a 3D point, they reproject it on the available views
and pool the corresponding 2D features. NeRFormer [38]
and ViewFormer [18] generalise these architectures by using
a transformer to pool information across views and along
rays [50]. However, these methods assume that the observed
object is rigid, or are limited to using a single view. We con-
tribute instead the first dynamic new-view synthesis method
that can use a sparse set of views as input.
Datasets for new-view synthesis. There are only a few real-life datasets that can be used to train models for category-specific new-view synthesis. Choi et al. [3] and GSO [7] introduced datasets of 2,000 real-world videos each. Each video shows around the object and has ground-truth depth. Objectron [1] introduced 15,000 videos, but only some of them show the object from all sides. Objaverse [4] is a large-scale 3D dataset of over 800,000 synthetic models. Dove [53] provided a dataset consisting of a few long videos of birds shot from static cameras [53], but with comparatively little variety. The CO3D dataset [38] contains 19,000 fly-around videos of rigid objects of 50 common categories. Like CO3D, our new dataset CoP3D provides fly-around videos of the objects, with the difference that the pets deform over time, which allows to study new-view synthesis of deformable objects.

3. Common Pets in 3D: a new dataset

Our new Common Pets in 3D (CoP3D) dataset contains 4,200 videos of cats and dogs of different breeds, captured under different viewpoint, camera motion, background and illumination, and moving in different ways. Videos show around the pets and are collected ‘in the wild’ via crowdsourcing. Compared to CO3D [39], which is collected in a similar manner, reconstruction is much more challenging because pets can move over time, sometimes suddenly.

CoP3D is designed as a benchmark for new-view synthesis. Hence, in addition to the videos, CoP3D contains intrinsic and extrinsic camera parameters and masks of the reconstructed objects. It is a collection of videos \( V_i = \{(I_j^i, M_j^i, P_j^i, t_j^i)\}_{j=1}^{N_{Vi}} \), each consisting of a sequence of \( N_{Vi} \) RGB frames \( I_j^i \in \mathbb{R}^{H \times W} \), masks \( M_j^i \in [0, 1]^{H \times W} \), cameras \( P_j^i \in \mathbb{R}^{4 \times 3} \) and time stamps \( t_j^i \in \mathbb{R}_+ \).

Data collection. Videos in CoP3D were collected by asking Amazon Mechanical Turk (AMT) workers to capture 360° videos of their pets using a smartphone. Each video was manually reviewed for quality and to ensure that pet owners acted responsibly, ensuring the safety of their pets, others, and themselves. The videos focus on pets and do not contain personal information such as human faces.

Camera reconstruction. The camera parameters \( P_j^i \) were reconstructed by using COLMAP [41], an off-the-shelf Structure-from-Motion software, on a uniformly-sampled subset of \( N_{Vi} = 200 \) frames per video. Video frames are rectified to account for non-linear distortion and the camera parameters are given as \( 4 \times 4 \) projection matrices.

COLMAP does not reconstruct cameras correctly for all videos. We use a semi-automated method to assign a camera reconstruction quality score to each video and consider only the best 2,966 videos for the benchmark (1,234 cat and 1,732 dog videos). Even so, we release the full set of 4,200 videos as future SFM improvements will likely make it possible to use many more for reconstruction. For camera scoring, we follow the CO3D approach and use a human-in-the-loop active learning scheme. In each active learning iteration, a human looks at some videos with the generated COLMAP camera tracks and assigns a binary label indicating the subjective correctness of the tracking. An SVM classifier is then trained to predict camera correctness given a set of features characterizing the reconstruction. The SVM is evaluated on all videos and the results are given back to the annotator, who then labels false positives and false negatives to improve the SVM’s decision boundary. In total, 1,000 videos of cats and dogs were manually labelled to train the SVM.

4. Few-shot dynamic new-view synthesis

We discuss first necessary background (section 4.1) and then our reconstruction method, Tracker-NeRF (section 4.2).

4.1. Background

Many recent new-view synthesis methods represent the 3D object or scene as a radiance field, i.e., a pair of functions \( \sigma = f^\sigma(x) \) and \( c = f^c(x, r) \) that map 3D points \( x \in \mathbb{R}^3 \) and viewing directions \( r \in S^2 \) to a correspond-
Figure 3. Tracker-NeRF reconstructs non-rigid 3D shape given few source views. It estimates the 3D motion of the shape by predicting source-view-specific offsets for each point on a projection ray emitted from the target image. The offset points are then projected to each source view and the image features and CSE descriptors are sampled. Tracker-NeRF then aggregates the information for each 3D point and predicts the point-specific colour, opacity, and surface embedding. The latter is then rendered using Emission Absorption rendering.

Learning a category-level prior using a neural field. Fitting a neural field to a small number of views (e.g., fewer than 20) is highly ambiguous and a good result can only be obtained by using a prior to compensate for the missing information. One way to do this is to learn an autoencoder, where the neural field \( f(x, r, z) = (c, \sigma) \) is a function of an additional latent code \( z \in \mathbb{R}^{D^z} \) predicted by an encoder function \( z((I_{src}, I_{tr}) )_{i=1}^{N_{src}} \) of the \( N_{src} \) available source images. The autoencoder is learned form a large dataset and captures implicitly the required prior.

Methods differ in the way the autoencoder is designed. Warp-Conditioned Embedding (WCE) [10] uses an autoencoder inspired by the geometry of image formation. Given a source image \( I_{src} \), a WCE \( z_{WCE}(x, I_{src}) \) pools information about the 3D point \( x \) by looking at its 2D projection:

\[
\begin{align*}
    z_{WCE}(x, I_{src}) &= \Psi_{CNN}(I_{src})[\pi_{pim}(x)],
\end{align*}
\]

where \( \Psi_{CNN}(I_{src}) \in \mathbb{R}^{D^x \times H \times W} \) is a CNN that extracts dense 2D features from the source image and \( \pi_{pim}(x) \) is the projection of the 3D point on the source camera \( I_{src} \). The code \( z_{WCE}(x, (I_{src}, I_{tr}) )_{i=1}^{N_{src}} \) of multiple source images is the concatenation of the mean and standard deviation of the WCEs of the individual images.

NeRFormer [39] extends WCE by further processing codes with a transformer network [50]. Given a pixel \( u \), it forms a \( N_{src} \times N_{src} \) grid of \( D^x \)-dimensional WCE tokens \( Z_{WCE}(x, I_{src}) \in \mathbb{R}^{D^x \times N_{src} \times N_{src}} \), spanning the \( N_{src} \) ray samples \( x \), and \( N_{src} \) source views \( I_{src} \). The neural field is implemented by a network \( f_{TR}(Z_{WCE}) = (c_j, \sigma_j)_{j=1}^{N_{src}} \) that predicts colors and opacities for all ray samples \( x \). It does so by alternating transformer layers that attend to tokens...
along the ray and source view dimensions, respectively.

4.2. Tracker-NeRF

Tracker-NeRF (TrackeRF) extends NeRFormer [39] to dynamic objects by: (i) predicting the trajectory of each rendered 3D point in order to accommodate for the non-rigid deformation of the object, and (ii) leveraging the Canonical Surface Embedding [31] to help establish correspondences between different videos and object instances (see fig. 3).

Time Warp-Conditioned Embedding (TWCE). Both NeRF-WCE and NeRFormer assume a rigid object with a fixed pose across source frames $I_{src}$. Given a 3D point $x$ along the ray $r_u$, a Time Warp-Conditioned Embedding (TWCE) extends the WCE $z_{WCE}(x, I_{src})$ to also encode the timestamps $t^g$ and $t^r$ of the source and target images:

$$z_{TWCE}(x, I_{src}) = [z_{WCE}(x, I_{src}), \gamma(t^g), \gamma(t^r)],$$  \hspace{1cm} (2)

where $\gamma$ is the usual harmonic encoding. Adding the timestamps allows the neural field to ‘sense’ time and model dynamic objects in the video.

Modelling non-rigid deformations. A shortcoming of the TWCE embedding of eq. (2) is that the image features are still sampled according to eq. (1), which fails to account for the motion of the object. In TrackeRF, we propose to explicitly compensate for this motion. We do so by predicting the location of the 3D point $x$ at time $t^r$ as:

$$\bar{x}(t^r) = x + \delta(x, t^g, t^r),$$  \hspace{1cm} (3)

where $\delta(x, t^g, t^r) \in \mathbb{R}^3$ is the displacement of $x$ from the target time $t^g$ to the source time $t^r$, also known as scene flow. The scene flow $\delta(x, t^g, t^r)$ is predicted by a modified NeRFormer network

$$\{\delta(x_j, t^g, t^r)\}_{j=1}^{N_S} = D_{TR}(Z_{r_u}^{TWCE}),$$  \hspace{1cm} (4)

which takes as input the grid of TWCE tokens $Z_{TWCE}^{r_u} = [z_{TWCE}(x_j, I_{t^r})] \in \mathbb{R}^{D_x \times N_S \times N_m}$.

Tracker-NeRF. The network $D_{TR}$ still takes as input the TWCE pooled from potentially incorrect 2D locations; its goal is to estimate such errors and ‘resolve’ them by computing the adjusted 3D points $\bar{x}(t^r)$. The adjusted points are then used in a vanilla NeRFormer network $f_{TR}$ to compute colors and opacities. This is done by evaluating the function

$$\{\bar{c}_{r_u}^j, \sigma_{r_u}^j\}_{j=1}^{N_S} = f_{TR}(Z_{r_u}^{TWCE}),$$  \hspace{1cm} (5)

which takes as input the resampled tokens $Z_{TWCE}^{r_u} = [z_{TWCE}(x_j(t^r), I_{t^r})] \in \mathbb{R}^{D_x \times N_S \times N_m}$. Intuitively, by using the offset 3D points $x_j(t^r)$, the TWCEs pool information from pixels in the source views that are in proper correspondence to the target 3D point.

Note that TrackeRF differs significantly from prior works that also model point trajectories [21, 26, 34, 54]: while the latter retrain the deformation model from scratch for every reconstructed scene, TrackeRF trains a single model on a large dataset of videos, and later predicts deformations in a feed-forward manner on new videos without retraining, using an auto-encoder.

Flow-consistent TrackeRF. Since learning the scene flow $f^g = \delta(x, t^g, t^r)$ is generally ill-posed, we further constrain it to match the 2D optical flow via the following loss:

$$L_{flow} = \sum_{u \in M_{tgt \rightarrow src}} \sum_{j \in u^w} w(u_j) e(x_j, t^r),$$  \hspace{1cm} (6)

$$e(x_j, t^r) = \|u + F_{tgt \rightarrow src}(u) - \pi_{src}(\bar{x}_j(t^r))\|_1,$$  \hspace{1cm} (7)

where $F_{tgt \rightarrow src}$ denotes RAFT [46] optical flow predictions, mapping pixels from the target frame $I_{tgt}$ to their corresponding 2D locations in the source frame $I_{src}$, and $|a|_\epsilon = \epsilon \cdot \left(\frac{1 + |a|^2}{\epsilon^2} - 1\right)$ is the soft Huber norm with cut-off threshold $\epsilon = 0.001$.

The loss is computed only for the pixels $u \in M_{tgt \rightarrow src}$ that belong to the foreground ($M_{tgt}(u) = 1$) and for which the optical flow is reliable. Reliability is computed by comparing the forward and backward flow, and masking pixels where the sum lands outside $\epsilon$ pixels from the original location [12]. Furthermore, the factor $w(u_j)$ restricts the loss to only the 3D points $x_j$ that approximately lie on the surface of the object and that are visible (and thus contribute to the optical flow). Finally, we stop gradient backpropagation from $L_{flow}$ to $w(x_j)$ as we observed this to improve convergence.

Canonical Surface Embeddings. We can further help the model by incorporating category-specific information captured by the Canonical Surface Embedding (CSE) of [31]. A CSE assigns to each pixel $u$ of the object an embedding vector that uniquely identifies the corresponding point on the surface of the object (also known as a canonical map). We use CSE models pre-trained for cats and dogs, respectively.

| Method                  | PSNR   | LPIPS  | IoU    | $\ell_{RGB}^*$  |
|-------------------------|--------|--------|--------|-----------------|
| SRN+AD                  | 17.2   | 0.24   | 0.78   | 0.27            |
| SRN+TWCE                | 16.8   | 0.19   | 0.75   | 0.29            |
| TimeNeRF                | 17.3   | 0.18   | 0.72   | 0.20            |
| NeRF+TWCE               | 17.3   | 0.19   | 0.73   | 0.46            |
| NeRFormer+TWCE          | 18.6   | 0.17   | 0.82   | 0.21            |
| NSFF                    | 20.2   | 0.17   | —      | 0.19            |
| TrackeRF (ours)         | 21.4   | 0.15   | 0.91   | 0.17            |
| FT+NeRF+TWCE            | 17.7   | 0.19   | 0.82   | 0.30            |
| FT+NeRFormer+TWCE       | 20.5   | 0.15   | 0.88   | 0.20            |
| FT+TrackeRF (ours)      | 23.1   | 0.13   | 0.91   | 0.17            |

Table 1. Many-Shot Single-Scene Reconstruction (MSSSR) on CoP3D. Models prefixed by the string FT were pre-trained on the FSCR task and later separately fine-tuned to each scene of the MSSSR task (best/2nd best).
Table 2. Few-Shot Category Reconstruction (FSCR) results on CoP3D. Metrics are averaged over both categories of CoP3D (cats and dogs). We report: (a) average results over the whole dataset, and (b) PSNR depending on the number of source views $N_{src}$ (best/2nd best).

| Method            | train-unseen PSNR | train-unseen PSNR | test-unseen PSNR | test-unseen PSNR |
|-------------------|-------------------|-------------------|------------------|------------------|
|                   | $I_1^{RGB}$       | $I_1^{RGB}$       | $I_1^{RGB}$      | $I_1^{RGB}$      |
| TrackeRF (ours)   | 19.1              | 0.16              | 0.33             | 0.80             |
| NeRFFormer+TWCE   | 16.3              | 0.19              | 0.39             | 0.73             |
| NeRF+TWCE [10]    | 14.6              | 0.20              | 0.48             | 1.62             |
| SRN+TWCE          | 13.9              | 0.18              | 0.52             | 0.53             |
| SRN+AD [43]       | 15.5              | 0.19              | 0.40             | 0.66             |

Table 3. Ablation study of Tracker-NeRF on the test-unseen subset of CoP3D few-view reconstruction (best in bold).

| predict $\delta$ | $L_{\text{flow}}$ | $L_{\text{CSE}}$ | PSNR | LPIPS | $I_1^{RGB}$ | IoU |
|------------------|-------------------|------------------|------|-------|-------------|-----|
| $\times$         | $\times$          | $\times$         | 16.6 | 0.18  | 0.36        | 0.76|
| $\checkmark$     | $\times$          | $\times$         | 17.3 | 0.18  | 0.35        | 0.77|
| $\times$         | $\checkmark$      | $\times$         | 17.6 | 0.18  | 0.35        | 0.80|
| $\checkmark$     | $\checkmark$      | $\times$         | 17.8 | 0.18  | 0.35        | 0.77|
| $\checkmark$     | $\checkmark$      | $\checkmark$     | 18.9 | 0.17  | 0.32        | 0.80|
| $\checkmark$     | $\checkmark$      | $\checkmark$     | 19.6 | 0.17  | 0.31        | 0.82|

The scene flow vectors $\delta$. Second, similar to [56], we task the model with predicting the canonical surface embeddings for the target view by considering an extended field $f^{\text{CSE}}(x, r, z) = (c, \sigma, C)$ which assigns to each 3D point $x$ a CSE vector $C \in \mathbb{R}^{D_{\text{CSE}}}$ in addition to a color $c$ and an opacity $\sigma$. This field is supervised by rendering: in addition to the RGB colors, we also render the corresponding CSE vectors, and minimize the CSE rendering loss $L_{\text{CSE}}$:

$$L_{\text{CSE}} = \| M^{\text{gt}} \odot (\Psi_{\text{CSE}}(I^{\text{gt}}) - \bar{C}^{\text{gt}}_{\text{CSE}}) \|_{\epsilon},$$

where $\bar{C}^{\text{gt}}_{\text{CSE}} \in \mathbb{R}^{D_{\text{CSE}} \times H \times W}$ are the rendered CSE embeddings generated with the same Emission-Absorption model used to render colors $I^{\text{gt}}$, and $\Psi_{\text{CSE}}(I^{\text{gt}}) \in \mathbb{R}^{D_{\text{CSE}} \times H \times W}$ are embeddings extracted from the target image $I^{\text{gt}}$ by the pretrained CSE network $\Psi_{\text{CSE}}$. We use an off-the-shelf CSE pretrained model which can be found here.
Optimization. We minimize the total loss $\lambda_{\text{photo}} L_{\text{photo}} + \lambda_{\text{mask}} L_{\text{mask}} + \lambda_{\text{flow}} L_{\text{flow}} + \lambda_{\text{CSE}} L_{\text{CSE}}$ (with $\lambda_{\text{photo}} = 1, \lambda_{\text{mask}} = 1, \lambda_{\text{flow}} = 1000, \lambda_{\text{CSE}} = 10$) using Adam optimizer with an initial learning rate of $5 \times 10^{-4}$, which is decayed by a factor of 10 whenever the total loss plateaus. During training, we render into a known target view and randomly sample between 5 and 25 source views.

5. Experiments

Data. For evaluation, the frames in the videos are split into four sets, similar to CO3D [38]. 50 videos in each category (cats and dogs) are selected at random as test videos and the others are used as training videos. Frames in each video (training and test) are further split by considering contiguous blocks of 15 frames as known, interleaved by blocks of 5 frames as unseen. The known frames are used as input to reconstruction algorithms whereas the unseen frames are only used for evaluation, providing the unseen camera parameters and timestamps, but withholding the images and masks, which must be predicted.

Tasks. We defined two benchmark tasks, also similar to CO3D. The first task, Many-Shots Single-Scene Reconstruction (MSSSR), is analogous to the setup explored in new-view synthesis approaches like NeRF [28]. The goal is to reconstruct the unseen frames from all the known frames in each of 10 test videos for each category (cats and dogs). We consider only 10 videos for evaluation since in the MSSSR setting a separate model is ‘overfitted’ to each video, which is expensive. The other task, Few-Shots Category Reconstruction (FSCR), is instead aimed at learning a category-specific prior and using it for reconstructing objects from a small number of views. In the training phase, the model is given access to the train-seen subset of video frames (including masks, cameras and timestamps). In the testing phase, the model receives $N_{\text{src}} \in \{5, 10, 15, 20, 25\}$ known source video frames $\{(I_{\text{src}}^i, M_{\text{src}}^i, P_{\text{src}}^i, t_{\text{src}}^i)\}_{i=1}^{N_{\text{src}}}$ from a test video. It also receives the camera parameters $P_{\text{tgt}}$ and timestamp $t_{\text{tgt}}$ of an unseen target frame and is tasked with predicting the target image $\hat{I}_{\text{tgt}}$ and mask $\hat{M}_{\text{tgt}}$.

Evaluation. We test the quality of reconstructed unseen images using several metrics: the Peak Signal-to-Noise Ratio (PSNR), the $\ell_1$ distance ($\ell_1^{\text{RGB}}$), the Learned Perceptual Image Patch Similarity (LPIPS [60]) between the predicted and ground-truth images, and the Intersection-over-Union (IoU) between the predicted and ground-truth object masks.
5.1. Many-Shots Single-Scene Reconstruction

**Baselines.** For the MSSSR task, we consider the following baselines: Scene Representation Networks (SRN) [43], SRN using the Time-Warp-Conditioned Embeddings (SRN-TWCE) [10, 43], time-conditioned variant of Neural Radiance Fields [28], which extends NeRF by appending the harmonic encoding of a frame’s timestamp to the spatial encoding of the 3D point coordinates (Time-NeRF), NeRF with Time-Warp-Conditioned Embedding (NeRF-TWCE), NeRFormer [39] using TWCE instead of WCE (NeRFormer-TWCE), and Neural Scene Flow Fields (NSFF) [21].

**Results.** Quantitative and qualitative results are given in table 1 and fig. 4, respectively. TrackeRF outperforms all baselines quantitatively, and the renders look visually superior and less blurry, in comparison.

5.2. Few-Shots Category Reconstruction

**Baselines.** For the FSCR problem, we consider the same baselines as before but only if they can be adapted to this setting. Specifically, we consider: NeRFormer-TWCE, NeRF-TWCE, SRN-TWCE, and our TrackeRF. We also consider SRN+AD from [38], which conditions an SRN model on an additional latent code which is optimized as a scene-specific free parameter together with the rest of the network weights.

**Results.** Quantitative and qualitative results are given in table 2 and fig. 5, respectively. Once more, TrackeRF outperforms all other baseline methods, this time by a wider margin of improvements. This is compatible with the fact that few-shot reconstruction depends more on the quality of the learned 3D prior. Qualitatively, TrackeRF produces significantly smoother spatial and temporal interpolations.

**Ablations.** In table 3, we evaluate the contribution of each of the TrackeRF’s components for reconstructing dynamic objects, by switching them off and measuring the change in reconstruction accuracy in the FSCR setting. The latter shows that there is a significant performance drop when any of the components is removed, which justifies the design choices made. We further visualize the rendered continuous surface embeddings (CSEs) in from different viewpoints fig. 6. This shows that TrackeRF can learn to interpolate the CSEs smoothly and learn correct dense correspondences across different views of the animals.

5.3. Single-scenes with pre-training

Finally, we test if models for MSSSR, which are ‘overfitted’ to a single video at a time, can benefit from category-centric pretraining of FSCR. Specifically, we follow the same evaluation procedure as in the MSSSR case, but the weights of each model are initialized by pre-training in the FSCR setting on the appropriate category, and then finetuned on the known frames of the single-scene with a learning rate of $10^{-5}$. The results in table 2 and fig. 5 for the FT+ models show that indeed pre-training improves results across the board, further demonstrating how CoP3D can be used to learn category-specific 3D priors by simply fine-tuning the category-centric model on to the new video.

6. Conclusions

We have introduced Common Pets in 3D, a new large-scale dataset to explore the problem of new-view synthesis of deformable objects. The dataset consists of 4,200 videos of cats and dogs collected in the wild using smartphone devices. Our new dataset supports research in 4D reconstruction from casually-recorded videos, which can be very impactful in applications such as VR & AR. The data allows to learn 4D reconstruction category-priors, that are useful for reconstructing non-rigid objects with better visual quality and from less data. We further introduce a new method, Tracker-NeRF, which learns a prior on the deformation of objects in videos. We have further demonstrated the benefit of such learned category priors by improving the quality and training time of single-sequence reconstruction by first pre-training the model on the category-level reconstruction task, and then fine-tuning on the new sequence.
References

[1] Ahmadyan, A., Zhang, L., Wei, J., Ablavatski, A., Grundmann, M.: Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. Proc. CVPR (2021) 3

[2] Chen, X., Zheng, Y., Black, M.J., Hilliges, O., Geiger, A.: SNARF: differentiable forward skinning for animating non-rigid neural implicit shapes. arXiv:cs abs/2104.03953 (2021) 2

[3] Choi, S., Zhou, Q.Y., Miller, S., Koltun, V.: A Large Dataset of Object Scans (2016), http://arxiv.org/abs/1602.02481 3

[4] Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. arXiv preprint arXiv:2212.08051 (2022) 3

[5] Du, Y., Zhang, Y., Yu, H.X., Tenenbaum, J.B., Wu, J.: Neural radiance field for 3D view synthesis and video processing. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14304–14314. IEEE Computer Society (2021) 2

[6] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022) 2

[7] GoogleResearch: Google scanned objects (September), https://fuel.ignitionrobotics.org/1.0/GoogleResearch/fuel/collections/GoogleScannedObjects 3

[8] Henzler, P., Mitra, N.J., Ritschel, T.: Escaping plato’s cave using adversarial training: 3D shape from unstructured 2D image collections. In: Proc. ICCV (2019) 4

[9] Henzler, P., Reizenstein, J., Labatut, P., Shapovalov, R., Ritschel, T., Vedaldi, A., Novotny, D.: Unsupervised learning of 3D object categories from videos in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4700–4709 (2021) 2

[10] Henzler, P., Reizenstein, J., Labatut, P., Shapovalov, R., Ritschel, T., Vedaldi, A., Novotny, D.: Unsupervised learning of 3D object categories from videos in the wild. Proc. CVPR (2021) 2, 4, 6, 8

[11] Jang, W., Agapito, L.: Codenerf: Disentangled neural radiance fields for object categories. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12949–12958 (2021) 2

[12] Jeong, J., Lin, J.M., Porikli, F., Kwak, N.: Imposing consistency for optical flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3181–3191 (2022) 5

[13] Jiang, W., Yi, K.M., Samei, G., Tuzel, O., Ranjan, A.: Neuman: Neural human radiance field from a single video. arXiv preprint arXiv:2203.12575 (2022) 2

[14] Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: Proc. ECCV (2018) 2

[15] Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. Advances in neural information processing systems 30 (2017) 2

[16] Kirillov, A., Wu, Y., He, K., Girshick, R.: Pointrdf: Image segmentation as rendering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9799–9808 (2020) 12

[17] Kokkinos, F., Kokkinos, I.: Learning monocular 3D reconstruction of articulated categories from motion. In: IEEE Conference on Computer Vision and Pattern Recognition (2021). https://doi.org/10.1109/cvpr46437.2021.00178, http://arxiv.org/abs/2103.16352 2

[18] Kulhánek, J., Derner, E., Sattler, T., Babuška, R.: Viewformer: Nerf-free neural rendering from few images using transformers. arXiv preprint arXiv:2203.10157 (2022) 2

[19] Li, X., Liu, S., Kim, K., Mello, S.D., Jampani, V., Yang, M.H., Kautz, J.: Self-supervised single-view 3D reconstruction via semantic consistency. In: European Conference on Computer Vision. pp. 677–693. Springer (2020) 2

[20] Li, X., Liu, S., Kim, K., Mello, S.D., Jampani, V., Yang, M., Kautz, J.: Self-supervised single-view 3D reconstruction via semantic consistency. In: Proc. ECCV (2020) 2

[21] Li, Z., Yu, F., Zollhöfer, M., Rhodin, H.: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2021), https://arxiv.org/abs/2011.13084 2, 5, 8

[22] Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. NeurIPS (2020) 1

[23] Liu, L., Habermann, M., Rudnev, V., Sarkar, K., Gu, J., Theobalt, C.: Neural actor: Neural Free-view Synthesis of Human Actors with Pose Control. ACM Transactions on Graphics 40(6), 1–16 (2021). https://doi.org/10.1145/3478513.3480528 2

[24] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM Transactions on Graphics 38(4) (2019). https://doi.org/10.1145/3306346.3323020 1, 2

[25] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019) 2

[26] Luo, X., Huang, J., Szeliski, R., Matzen, K., Kopf, J.: Consistent video depth estimation. ACM Trans. Graph. 39(4), 71 (2020) 5

[27] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: European Conference on Computer Vision (2020), http://arxiv.org/abs/2003.08934 1, 2

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
[56] Yang, G., Vo, M., Neverova, N., Ramanan, D., Vedaldi, A., Joo, H.: BANMo: Building Animatable 3D Neural Models from Many Casual Videos (2021), http://arxiv.org/abs/2112.12761

[57] Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. In: Proc. NeurIPS (2020)

[58] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. arXiv.cs abs/2012.02190 (2020)

[59] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural Radiance Fields from One or Few Images (2020), http://arxiv.org/abs/2012.02190

[60] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)