Common Objects in 3D: 
Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction

Jeremy Reizenstein¹  Roman Shapovalov¹  Philipp Henzler²  Luca Sbordone¹  
Patrick Labatut¹  David Novotny¹  

¹Facebook AI Research  ²University College London  

https://github.com/facebookresearch/co3d

1. Introduction

Recently, the community witnessed numerous advances in deeply learning to reconstruct category-centric 3D models. While a large variety of technical approaches was proposed [56, 62, 13, 25, 26, 46, 8], they are predominantly trained and benchmarked either on synthetic data [6], or on real datasets of specific object categories such as birds [62] or chairs [37]. The latter is primarily a consequence of a lack of relevant real-world datasets with 3D ground truth.

Our main goal is therefore to collect a large-scale open real-life dataset of common objects in the wild annotated...
with 3D ground truth. While the latter can be collected with specialized hardware (turn-table 3D scanner, dome [29]), it is challenging to reach the scale of synthetic datasets [6] comprising thousands of instances of diverse categories.

Instead, we devise a photogrammetric approach only requiring object-centric multi-view RGB images. Such data can be effectively gathered in huge quantities by means of crowd-sourcing “turn-table” videos captured with smartphones, which are nowadays a commonly owned accessory. The mature Structure-from-Motion (SfM) framework then provides 3D annotations by tracking cameras and reconstructs a dense 3D point cloud capturing the object surface.

To this end, we collected almost 19,000 videos of 50 MS-COCO categories with 1.5 million frames, each annotated with camera pose, where 20% of the videos are annotated with a semi-manually verified high-resolution 3D point cloud. As such, the dataset exceeds alternatives [9, 1, 21] in terms of number of categories and objects.

Our work is an extension of the dataset from [26]. Here, we significantly increase the dataset size from less than 10 categories to 50 and, more importantly, conduct a human-in-the-loop check ensuring reliable accuracy of all cameras. Finally, the dataset from [26] did not contain any point cloud annotations, the examples of which are in fig. 1.

We also propose a novel NerFormer model that, given a small number of input source views, learns to reconstruct object categories in our dataset. NerFormer mates two of the main workhorses of machine learning and 3D computer vision: Transformers [65] and neural implicit rendering [43]. Specifically, given a set of 3D points along a rendering ray, features are sampled from known images and stacked into a tensor. The latter is in fact a ray-depth-ordered sequence of sets of sampled features which admits processing with a sequence-to-sequence Transformer. Therefore, by means of alternating feature pooling attention and ray-wise attention layers, NerFormer learns to jointly aggregate features from the source views and raymarch over them.

Importantly, NerFormer outperforms a total of 14 baselines which leverage the most common shape representations to date. As such, our paper conducts one of the first truly large-scale evaluations of learning 3D object categories in the wild.

2. Related Work

In this section we review current 3D datasets and related methods in the areas of single-image reconstruction, generative modelling and novel-view synthesis.

3D object datasets The main enabler of early works in 3D reconstruction was the synthetic ShapeNet [6] dataset. Pascal3D [72] introduced a real world dataset providing pose estimation for images, but only approximate 3D models. Choi et al. [9] provide a large set of realobject-centric RGB-D videos, however only a small subset is annotated with 3D models and cameras. Increasing the number of categories and objects, Objectron [1] contains object-centric videos, object/camera poses, point clouds, surface planes and 3D bounding boxes. Unfortunately, only a limited number of object-centric videos cover full 360 degrees. Our dataset further increases the number of categories by a factor of 5 and covers the full 360 degree range. Requiring 3D scanners, GSO [21] provides clean full 3D models including textures of real world objects. Due to the requirement of 3D scanning, it contains less objects. A detailed comparison of the aforementioned datasets is presented in tab. 1.

3D reconstruction A vast amount of methods studied fully supervised 3D reconstruction from 2D images making use of several different representations: voxel grids [10, 18], meshes [19, 67], point clouds [14, 73], signed distance fields, [49, 2] or continuous occupancy fields [42, 8, 17, 16].

Methods overcoming the need for 3D supervision are based on differentiable rendering allowing for comparison of 2D images rather than 3D shapes [51, 63, 31]. Generating images from meshes was achieved via soft rasterization in [32, 40, 61, 30, 7, 36, 77, 20, 67]. Volumetric representations are projected to 2D via differentiable raymarching [25, 15, 41, 43, 39, 55] or in a similar fashion via sphere tracing for signed distance fields [45, 74]. [28] introduce differentiable point clouds. Another line of work focuses on neural rendering, i.e. neural networks are trained to approximate the rendering function [44, 56]. [35, 34] map pixels to object-specific deformable template shapes. [47, 48] canonically align point clouds of object-categories in an unsupervised fashion, but do not reason about colour. Exploiting symmetries and reasoning about lighting [71], compose appearance, shape and lighting.

Similar to us, [52, 75, 26, 60] utilize per-pixel warp-conditioned embedding [26]. In contrast to our method, multi-view aggregation is handled by averaging over encodings which is prone to noise. Instead, we learn the aggregation by introducing the NerFormer module. Finally, a very recent IBRNet [68] learns to copy existing colors from known views in an IBR fashion [24, 5], whereas our method can hallucinate new colors, which is crucial since we leverage far fewer source views (at most 9).

| Table 1: Common Objects in 3D compared to alternatives. The “pcl” abbreviation stands for “point clouds”. |
|---|---|---|---|---|---|---|---|
| Dataset | [6] | [72] | [9] | [1] | [21] | [26] | Ours |
| Real | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| # Categories | 55 | 12 | 9 | 9 | NA | 7 | 50 |
| # Objects | 51k | 36k | 2k | 15k | 2k | 2k | 19k |
| 3D GT | full approx. depth pcl box | full pcl pcl | Multi-view | full none | limited full | full full |

The “pcl” abbreviation stands for “point clouds”.

10902
3. Common Objects in 3D

In this section we describe the dataset collection process.

**AMT video collection** In order to scale the collection of object-centric videos, we crowd-sourced it on Amazon Mechanical Turk (AMT). Each AMT task asks a worker to select an object of a given category, place it on a solid surface and take a video where they keep the whole object in view while moving full circle around it. We pre-selected 50 MS-COCO [38] categories (listed in fig. 2) comprising stationary objects that are typically large enough to be reconstructed. The workers were instructed to avoid actions that would hinder the ensuing reconstruction stage such as abrupt movements leading to motion blur. Each video was reviewed to ensure that it fulfills the requirements.

**Generating 3D ground-truth** As explained below, we use off-the-shelf software to produce object masks, camera tracking, and 3D reconstructions for each video. We then semi-automatically filter out poor reconstructions.

1) **Sparse reconstruction** Given the set of valid object-centric videos, we reconstruct the extrinsic (3D location and orientation) and intrinsic (calibration) properties of the cameras that captured the videos. To this end, each video is first converted into a time-ordered sequence of images \( V = \{ \mathbf{I}^i \mid \mathbf{I}^i \in \mathbb{R}^{H \times W} \}_{i=1}^{N_V} \) by extracting \( n_t = 100 \) frames uniformly spaced in time. The frames are then fed to the COLMAP SfM pipeline [53] which annotates each image with camera projection matrices \( \mathcal{P} = (\mathbf{P}^i \mid \mathbf{P}^i \in \mathbb{R}^{4 \times 4})_{i=1}^{N_V} \). Fig. 3 shows example camera tracks together with estimated sparse scene geometries.

2) **Object segmentation** We segment the object in each image \( \mathbf{I}^i \) with PointRend [33], state-of-the-art instance segmentation method, resulting in a sequence of soft binary masks \( \mathcal{M} = (\mathcal{M}^i \mid \mathcal{M}^i \in [0, 1]^{H \times W})_{i=1}^{N_V} \) per video. Note that, while masks aid further dense reconstruction, we have not used them for camera tracking which is typically anchored to the background regions.

3) **Semantic dense reconstruction** Having obtained the camera motions and segmentations, we now describe the process of annotating the captured objects with a 3D surface. We first execute the multi-view stereo (MVS) algorithm of COLMAP [54] to generate per-frame dense depth maps \( (D^i \mid D^i \in \mathbb{R}^{H \times W})_{i=1}^{N_V} \). We then run COLMAP’s point cloud fusion algorithm, which back-projects the depth values masked with \( \mathcal{M} \) and retains the points that are consistent across frames, to get a point cloud \( \mathcal{P}(\mathcal{V}) = \{\mathbf{x}_i\}_{i=1}^{N_R} \). Example dense point clouds are visualized in fig. 3.

4) **Labelling reconstruction quality with Human-in-the-loop.** Since our reconstruction pipeline is completely automated, any of the aforementioned steps can fail, resulting in unreliable 3D annotations. We thus incorporate a semi-manual check that filters inaccurate reconstructions.

To this end, we employ an active learning [11] pipeline which cycles between: a) manually labelling the point-cloud and camera-tracking quality; b) retraining a “quality” SVM classifier; and c) automatically estimating the shape/tracking quality of unlabelled videos. Details of this process are deferred to the supplementary.

5) **The dataset** The active-learned SVM aids the final dataset filtering step. As accurate camera annotations are crucial for the majority of recent 3D category reconstruction methods, the first filtering stage completely removes all scenes with camera tracking classified as “inaccurate” (18% of all videos). While all videos that pass the camera check are suitable for training, the scenes that pass both camera and point cloud checks (30% of the videos with accurate cameras) comprise the pool from which evaluation videos are selected. Note, a failure to pass the point cloud check does not entail that the corresponding scene is irreconstructible and should therefore be removed from training – instead this merely implies that the MVS method [54] failed, while other alternatives (see sec. 5) could succeed.

Fig. 2 summarizes the size of Common Objects in 3D. Reconstructing a single video took on average 1h 56 minutes with the majority of execution time spent on GPU-accelerated MVS and correspondence estimation. This amounts to the total of 43819 GPU-hours of reconstruction time distributed to a large GPU cluster.

4. Learning 3D categories in the wild

Here, we describe the problem set, give an overview of implicit shape representations, and explain our main techni-
Figure 3: 3D ground truth for Common Objects in 3D was generated with active learning. Videos are first annotated with cameras and dense point clouds with COLMAP [53]. Given several reconstruction metrics and manual binary annotations (“accurate”/“inaccurate”) of a representative subset of reconstructions, we train an SVM that automatically labels all videos.
Warp-conditioned embedding ages of unaligned scenes can generate vastly different 3D. similar 3D shapes, in the latter case, similarly looking im-
is possible since similar images of aligned scenes generate unlike in the former case where color-based shape inference
jects are arbitrarily placed in the 3D space. This is because, such approach is infeasible in real world settings where ob-
rigidly aligned, it has been recently shown in \[ [\text{56}] \].

\[ e(x^i_u, r_u, z) \text{ weighted by the emission-absorption product} \]
\[ w_i = \left( \prod_{j=0}^{i-1} T_j \right) (1 - T_i) \text{ with } T_i = \exp(-\Delta f_o(x^i_{src}, z)). \]

4.2. Latent shape encoding \( z \)

A crucial part of a category-centric reconstructor is the latent embedding \( z \). Early methods \([18, 70, 58, 61, 40]\) predicted a global scene encoding \( z_{global} = \Phi_{CNN}(I_{src}) \) with a deep convolutional network \( \Phi_{CNN} \) that solely analyzed the colors of source image pixels. While this approach was successful for the synthetic ShapeNet dataset where shapes are rigidly aligned, it has been recently shown in \([26, 75]\) that such approach is infeasible in real world settings where objects are arbitrarily placed in the 3D space. This is because, unlike in the former case where color-based shape inference is possible since similar images of aligned scenes generate similar 3D shapes, in the latter case, similarly looking images of unaligned scenes can generate vastly different 3D.

Warp-conditioned embedding To fix the latter, \([26]\) proposed Warp-Conditioned Embedding (WCE): given a world coordinate point \( x \) and a source view \( I_{src} \) with camera \( P_{src} \), the warp-conditioned embedding \( z_{WCE} \in \mathbb{R}^{D_z} \),

\[ z_{WCE}(x, I_{src}, P_{src}) = \Psi_{CNN}(I_{src})[\pi_{pmm}(x)], \]

is formed by sampling a tensor of source image descriptors \( \Psi_{CNN}(I_{src}) \in \mathbb{R}^{D_z \times H \times W} \) at a 2D location \( \pi_{pmm}(x) \). Here, \( \pi_{pmm}(x) = P_{src}[x; 1] = d_u[u; 1] \) expresses perspective camera projection of a 3D point \( x \) to a pixel \( u \) with depth \( d_u \in \mathbb{R} \). Intuitively, since \( z_{WCE} \) is a function of the world coordinate \( x \), the ensuing implicit \( f \) can perceive the specific 3D location of the sampled appearance element in the world coordinates, which in turn enables \( f \) to learn invariance to rigid scene misalignment.

In the common case where multiple source views are given, the aggregate WCE \( z^*_{WCE}(x, \{I_{src}\}, \{P_{src}\}) \) is defined as a concatenation of the mean and standard deviation of the set of view-specific source embeddings \( \{z_{WCE}(x, I_{src}^i, P_{src}^i)\}_{i=1}^N \).

Boosting baselines with WCE The vast majority of existing methods for learning 3D categories leverage global shape embeddings \( z_{global} \), which renders them inapplicable to our real dataset. As WCE has been designed to alleviate this critical flaw, and because of its generic nature, in this paper we endow state-of-the-art category-centric 3D reconstruction methods with WCE in order to enable them for learning category-specific models on our real dataset.

To this end, we complement SRN \([56]\), NeuralVolumes \([41]\), and the Implicit Point Cloud (discussed later) with WCE. These extensions are detailed in the supplementary.

4.3. Attention is all you nerf

Limitations of WCE While \([26]\) has demonstrated that the combination of NeRF and WCE (termed NeRF-WCE) leads to performance improvements, our experiments indicated that a major shortcoming of NeRF-WCE is its inability to deal with cases where parts of the 3D domain are labelled with noisy WCE. This is because NeRF’s MLP \( f_{3DMLP} \) independently processes each 3D point and, as such, cannot detect failures and recover from them via spatial reasoning. The 3D deconvolutions of Neural Volumes \([41]\) are a potential solution, but we found that the method ultimately produces blurry renders due to the limited resolution of the voxel grid. The LSTM marcher of SRN \([56]\) is capable of spatial reasoning, which is however somewhat limited due to the low-capacity of the LSTM cell. Last but not least, a fundamental flaw is that simple averaging of the source-view WCE embeddings can suppress important features.

Our main technical contribution aims to alleviate these issues and follows a combination of two main design guidelines: 1) We replace the \( f_{3DMLP} \) with a more powerful architecture capable of spatial reasoning. 2) Instead of engineering the WCE aggregation function, we propose to learn it.
NerFormer As a solution, we propose to leverage the popular Transformer architecture [65]. Employing a sequence-to-sequence model is intuitive since the set of all WCE embeddings along a projection ray is in fact a depth-ordered descriptor sequence along the ray dimension, and an unordered sequence along the feature pooling dimension.

Formally, given a ray $r_u$, we define $Z^{r_u} \in \mathbb{R}^{N_z \times N_{\text{enc}} \times D_s}$ as a stacking of un-aggregated WCEs of all ray-points $x_{r_u}$:

$$Z^{r_u} = \left( \left[ \text{WCE}(x_{r_u}, I_{i}^{src}, P_{j}^{src}) \right]_{i=1}^{N_z} \right)_{j=1}^{N_s}.$$ 

The NerFormer module $f^{r_u}_{\text{TR}}(Z^{r_u}) = f^{\text{HEAD}}_{\text{TR}} \circ \text{TE}_L \circ \cdots \circ \text{TE}_1(Z^{r_u})$ replaces the feature backbone $f^{r_u}_{\text{MLP}}$ (sec. 4.1) with a series of $L$ 3D transformer modules $\text{TE}_i$ terminated by a weighted pooling head $f^{\text{HEAD}}_{\text{TR}}$ (fig. 4). Here, each 3D transformer module $\text{TE}_i$ is a pairing of Transformer Encoder [65] layers $\text{TE}^0(Z)$ and $\text{TE}^1(Z)$:

$$\text{MHA}^i(Z_l) = Z_l' = \text{LN}([\text{MHA}^i(Z_l, \text{dim}=d) + Z_l])$$

$$\text{TE}^0(Z_l) = \text{LN}(\text{MLP}_1(Z_l')) + Z_l'$$

$$\text{TE}^1(Z_l) = \text{TE}^0_1(\text{TE}^1(Z_l)) = Z_{l+1},$$  

MHA($Z, \text{dim}=d$) is a multi-head attention layer [65] whose attention vectors span the $d$-th dimension of the input tensor $Z$, MLP is a two-layer MLP with ReLU activation, and LN is Layer Normalization [3]. Intuitively, the alternation between ray and pooling attention of $\text{TE}^0(Z)$ and $\text{TE}^1(Z)$ facilitates learning to jointly aggregate WCE features from the source views and ray-march over them respectively.

Finally, $f^{r_u}_{\text{TR}}$ is terminated by a weighted pooling head $f^{\text{HEAD}}_{\text{TR}}$ that aggregates the second dimension of $Z_L$ output by the final $L$-th 3D transformer module $\text{TE}_L$: $f^{\text{HEAD}}_{\text{TR}}(Z_L) = \sum_{i=1}^{n_{\text{w}}} \omega_i(Z_L)Z_L[i,:,:] \in \mathbb{R}^{N_z \times D_s}$, where the weights $\omega_i \in [0,1]$, $\sum_i \omega_i = 1$ are output by a linear layer with softmax activation. We show $\omega_i$ in fig. 5.

![Figure 6: Single-scene new-view synthesis on Common Objects in 3D depicting a target image from the training video (left), and corresponding synthesized view generated by IPC, SRN [56], NV [41], IDR [74], NerF [43], and our NerFormer.](image)

<table>
<thead>
<tr>
<th>method</th>
<th>PSNR</th>
<th>LPIPS $\ell^\text{1/4}$</th>
<th>IoU</th>
<th>PSNR</th>
<th>LPIPS $\ell^\text{1/4}$</th>
<th>IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>NerFormer</td>
<td>23.3</td>
<td>0.17</td>
<td>0.40</td>
<td>0.96</td>
<td>0.21</td>
<td>0.60</td>
</tr>
<tr>
<td>NerF+WCE[26]</td>
<td>21.0</td>
<td>0.19</td>
<td>0.74</td>
<td>0.91</td>
<td>0.30</td>
<td>0.60</td>
</tr>
<tr>
<td>NerF[43]</td>
<td>21.6</td>
<td>0.17</td>
<td>0.38</td>
<td>0.95</td>
<td>0.26</td>
<td>0.64</td>
</tr>
<tr>
<td>NV[41]</td>
<td>22.2</td>
<td>0.20</td>
<td>0.91</td>
<td>0.91</td>
<td>0.30</td>
<td>0.59</td>
</tr>
<tr>
<td>NV+WCE</td>
<td>18.7</td>
<td>0.25</td>
<td>0.85</td>
<td>0.90</td>
<td>0.33</td>
<td>0.89</td>
</tr>
<tr>
<td>IDR[74]</td>
<td>18.5</td>
<td>0.15</td>
<td>0.82</td>
<td>0.92</td>
<td>0.38</td>
<td>0.73</td>
</tr>
<tr>
<td>IPC+WCE</td>
<td>13.9</td>
<td>0.25</td>
<td>1.39</td>
<td>0.83</td>
<td>0.22</td>
<td>0.60</td>
</tr>
<tr>
<td>IPC</td>
<td>13.8</td>
<td>0.25</td>
<td>1.58</td>
<td>0.83</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Single-scene new-view synthesis results on Common Objects in 3D comparing the baseline approaches [43, 41, 74, 56, 45, 67], IPC, their variants with Warp-conditioned Embeddings (+WCE) or Positional Embedding (+γ), and our NerFormer (the best / 2nd best result).

The final opacity and coloring functions of NerFormer are thus $f_o = f^{\text{HEAD}} \circ f^{\text{TR}}_{\text{FR}}$, $c_o = c^{\text{HEAD}} \circ f^{\text{TR}}_{\text{FR}}$ respectively, which are rendered with the EA raymarcher (sec. 4.1).

**Technical details** Training minimizes, with Adam (learning rate $5 \cdot 10^{-4}$), a sum of the RGB MSE error $\|I_{\text{tgt}} - I_{\text{fr}}\|^2$ and the binary cross entropy between the rendered alpha mask $M_{\text{fr}}$ and the ground truth mask $M_{\text{gt}}$. We iterate over batches comprising a randomly sampled target view and 1 to 9 source views of a training video until convergence.

**5. Experiments**

**Datasets** In order to evaluate and train reconstruction methods on our dataset, we split the 18,619 collected CO3D videos into 4 different sets as follows. For each of the 50 categories, we split its videos to a train-test set in 9:1 ratio. For each video, we further define a set of frames that are removed from the training set by randomly dividing each train video in a 8:2 ratio to 80 train-known training and 20 train-unseen holdout frames. test video frames are split according to the same protocol resulting in test-known and test-unseen sets. As all base-
the Warp-conditioned Embedding \[ SRN- \] implements an implicit learned LSTM renderer. Importantly, DVR Implicit surfaces 5.1. Evaluated methods 5.1.2. Single-scene reconstruction Voxel grids Neural Volumes (NV) [41] represents the State of the Art among voxel grid predictors. Similar to implicit methods, we also combine NV with WCE. Point clouds In order to compare with a point cloud-based method, we devised an Implicit Point Cloud (IPC) baseline which represents shapes with a colored set of 3D points, converts the set into an implicit surface and then renders it with the EA raymarcher. We note that IPC is strongly inspired by SynSin [30, 69] (see supplementary). Meshes We benchmark P3DMesh - the best-performing variant of PyTorch3D’s soft mesh rasterizer from [50] inspired by Pixel2Mesh [67], which deforms an initial spherical mesh template with a fixed topology with a series of convolutions on the mesh graph. 5.2. Single-scene reconstruction We first task the approaches to independently reconstruct individual object videos. More specifically, given a test video, every baseline is trained to reproduce the video’s test-known frames (by minimizing method-specific loss functions such as \( L^2 \) RGB error), and evaluated by comparing the renders from the given test-known camera viewpoints to the corresponding ground truth images/depths/masks. Since training on all ~2k test videos is prohibitively expensive (each baseline trains at least for 24 hours on a single GPU), we test on 40 randomly selected test videos (two from each of 20 random classes). Quantitative/qualitative results are in fig. 6 / Tab. 2. NeRFer is either the best or the second best across all metrics. NeRF+WCE is beaten by vanilla NeRF which suggests that the noisy WCE embeddings can hurt performance without NeRFer’s spatial reasoning. Interestingly, IDR’s realistic, but less detailed renders win in terms of LPIPS, but are inferior in PSNR. Furthermore, we observed a large
discrepancy between the train and test PSNR of γ/WCE-endowed SRN. This shows that, for the single-scene setting, increasing the model expressivity with WCE or γ can lead to overfitting to the training views.

5.3. Learning 3D Object Categories

Our main task is learning category-centric 3D models. In more detail, a single model is trained on all train-known frames of an object category and evaluated on 1000 randomly generated test samples from train-unseen and test-unseen sets of the category. Each train or test sample is composed of a single target image $I_{tgt}$ and randomly selected source views $\{I_{src}^{n}\}_{n=1}^9$, where $n_{src}$ is randomly picked from $\{1, 3, 5, 7, 9\}$. Since training a model for each of the 50 object categories is prohibitively expensive (each method takes at least 7 days to train), we chose a subset of 10 categories for evaluation.

**Baselines** For each method we evaluate the ability to represent the shape space of training object instances by turning it into an autodecoder [4] which, in an encoder-less manner, learns a separate latent embedding $z_{scene}(sequence_id)$ for each train scene as a free training parameter. Since autodecoders (abbreviated with the +AD suffix) only represent the train set, we further compare to all WCE-based methods which can additionally reconstruct test videos. Note that, as remarked in [26] and sec. 4.2, the alternative encoding $z_{global} = \Phi_{CNN}(I_{src})$ is deemed to fail due to the world coordinate ambiguity of the training SfM reconstructions.

**Results** Quantitative and qualitative comparisons are shown in tab. 3 and fig. 7 respectively. Besides average metrics over both test sets, we also analyze the dependence on the number of available source views, and on the difficulty of the target view. For the latter, test frames are annotated with a measure of their distance to the set of available source views, and then split into 3 different difficulty bins. Average per-bin PSNR is reported. The supplementary details the distance metric and difficulty bins.

While SRN+AD has the best performance across all metrics on test-known; on the test-unseen set, where SRN+AD is inapplicable, our NerFormer is the best for most color metrics. Among implicit methods, the SDF-based DVR and IDR are outperformed by the opacity-based NeRF, with both DVR+WCE and IDR+WCE failing to converge. This is likely because regressing SDFs is more challenging than classifying 3D space with binary labels opaque/transparent. Finally, we observed poor performance of P3DMesh, probably due to the inability of meshes to represent complicated real geometries and textures.

6. Conclusion

We have introduced Common Objects in 3D (CO3D), a dataset of in-the-wild object-centric videos capturing 50 object categories with camera and point cloud annotations.

We further contributed NerFormer which is a marriage between Transformer and neural implicit rendering that can reconstruct 3D object categories from CO3D with better accuracy than a total of 14 other tested baselines.

The CO3D collection effort still continues at a steady pace of ~500 videos per week which we plan to release in the near future.
References


[21] GoogleResearch. Google scanned objects, September. 2

[22] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. 2016. 4


[26] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shpaovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3d object categories from videos in the wild. arXiv, 2021. 1, 2, 4, 5, 6, 7, 8, 13


