Class-agnostic Reconstruction of Dynamic Objects from Videos

Zhongzheng Ren[*], **Xiaoming Zhao**[*], **Alexander G. Schwing**
University of Illinois at Urbana-Champaign
https://jason718.github.io/redo

Abstract

We introduce **REDO**, a class-agnostic framework to **RE**construct the **D**ynamic **O**bjects from RGBD or calibrated videos. Compared to prior work, our problem setting is more realistic yet more challenging for three reasons: 1) due to occlusion or camera settings an object of interest may never be entirely visible, but we aim to reconstruct the complete shape; 2) we aim to handle different object dynamics including rigid motion, non-rigid motion, and articulation; 3) we aim to reconstruct different categories of objects with one unified framework. To address these challenges, we develop two novel modules. First, we introduce a canonical 4D implicit function which is pixel-aligned with aggregated temporal visual cues. Second, we develop a 4D transformation module which captures object dynamics to support temporal propagation and aggregation. We study the efficacy of REDO in extensive experiments on synthetic RGBD video datasets SAIL-VOS 3D and DeformingThings4D++, and on real-world video data 3DPW. We find REDO outperforms state-of-the-art dynamic reconstruction methods by a margin. In ablation studies we validate each developed component.

1 Introduction

4D (3D space + time) reconstruction of both the geometry and dynamics of different objects is a long-standing research problem, and is crucial for numerous applications across domains from robotics to augmented/virtual reality (AR/VR). However, complete and accurate 4D reconstruction from videos remains a great challenge for mainly three reasons: 1) partial visibility of objects due to occlusion or camera settings (*e.g.*, out-of-view parts, non-observable surfaces); 2) complexity of the dynamics including rigid-motion (*e.g.*, translation and rotation), non-rigid motion (deformation caused by external forces), and articulation; and 3) variability within and across object categories.

Existing work addresses the above challenges by assuming complete visibility through a multi-view setting [33, 4], or by recovering only the observable surface rather than the complete shape of an object [57], or by ignoring rigid object motion and recovering only the articulation [59], or by building shape templates or priors specific to a particular object category like humans [47]. However, these assumptions also limit applicability of models to unconstrained videos in the wild, where these challenges are either infeasible or only met when taking special care during a video capture.

In contrast, we aim to study the more challenging unconstrained 4D reconstruction setting where objects may never be entirely visible. Specifically, we deal with visual inputs that suffer from: 1) *occlusion:* the moving occluder and self-articulation cause occlusion to change across time; 2) *cropped view:* the camera view is limited and often fails to capture the complete and consistent appearance across time; 3) *front-view only:* due to limited camera motion, the back side of the objects are often not captured at all in the entire video. Moreover, we focus on different dynamic object-types

[*]Indicates equal contribution

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

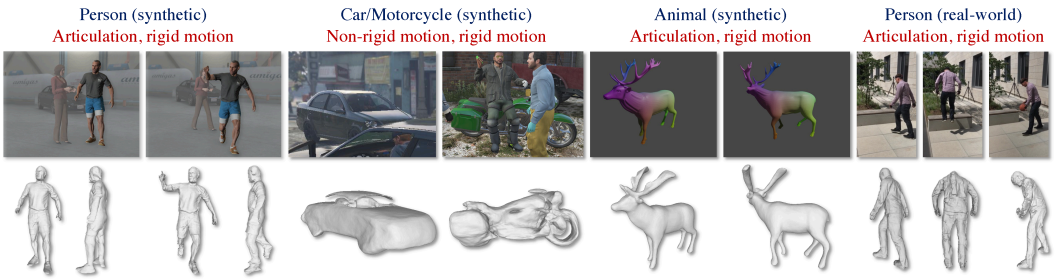


Figure 1: We present REDO, a 4D reconstruction framework that predicts the geometry and dynamics of various objects for a given video clip. Despite objects being either occluded (*e.g.*, car/motorcycle) or partially-observed, REDO recovers relatively complete and temporally smooth results.

with complex motion patterns. These objects could either move in 3D space, or be deformed due to external forces, or articulate themselves. Importantly, we aim for a class-agnostic reconstruction framework which can recover the accurate shape at each time-step.

To achieve this we develop REDO. As illustrated in Fig. 1, REDO predicts the shape of different objects (*e.g.*, human, animals, car) and models their dynamics (*e.g.*, articulation, non-rigid motion, rigid motion) given input video clips. Besides the RGB frames, REDO takes as input the depth map, masks of the objects of interest, and camera matrices. In practice these inputs are realistic as depth-sensors are increasingly prevalent [19, 78] and as segmentation models [44, 17] are increasingly accurate and readily available, *e.g.*, on mobile devices. If this data isn't accessible, off-the-shelf tools are applicable (*e.g.*, SfM [71], instance segmentation [26], video depth estimation [51]).

To address the partial visibility challenge introduced by occlusion or camera settings, REDO predicts a temporally coherent appearance in a canonical space. To ensure that the same model is able to reconstruct very different object types in a unified manner we introduce a pixel-aligned 4D implicit representation, which encourages the predicted shapes to closely align with the 2D visual inputs (§ 3.1). The visible parts from different frames of the video clip are aggregated to reconstruct the same object (§ 3.2). During inference, the reconstructed object in canonical space is propagated to other frames to ensure a temporally coherent prediction (§ 3.3).

REDO achieves state-of-the-art results on various benchmarks (§ 4). We first conduct experiments on two synthetic RGBD video datasets: SAIL-VOS 3D [27] and DeformingThings4D++ [42]. REDO improves over prior 4D reconstruction work [59, 70] by a great margin (+5.9 mIoU, -0.085 mCham., -0.22 mACD on SAIL-VOS 3D and +2.2 mIoU, -0.063 mCham., -0.047 mACD on DeformingThings4D++ over OFlow [59]). We then test on the real-world calibrated video dataset 3DPW [86]. We find that REDO generalizes well and consistently outperforms prior 4D reconstruction methods (+10.1 mIoU, -0.124 mCham., -0.061 mACD over OFlow). We provide a comprehensive analysis to validate the effectiveness of each of the introduced components.

2 Related work

In this section, we first discuss possible geometry representations. We then review fusion-based and learning-based 4D reconstruction approaches, followed by a brief introduction of Motion Capture methods. Lastly, we discuss works of dynamics modelling and related 4D reconstruction datasets.

Geometric representations. Representations to describe 3D objects can be categorized into two groups: discrete and continuous. Common discrete representations are voxel grids [22, 90, 8], octrees [68, 82], volumetric signed distance functions (SDFs) [10, 30, 56], point-clouds [1, 18, 67], and meshes [23, 34, 55, 38, 88]. Even though being widely used, these representations pose important challenges. Voxel grids and volumetric-SDFs can be easily processed with deep learning frameworks, but are memory inefficient [76, 52, 64]. Point-clouds are more memory efficient to process [65, 66], but do not contain any surface information and thus fail to capture fine details. Meshes are more expressive, but their topology and discretization introduce additional challenges.

To overcome these issues, continuous representations, *i.e.*, parametric implicit functions, are introduced to describe the 3D geometry of objects [7, 60, 53] and scenes [54, 73]. These methods are not constrained by discretization and can thus model arbitrary geometry at high resolution [70, 81]. In

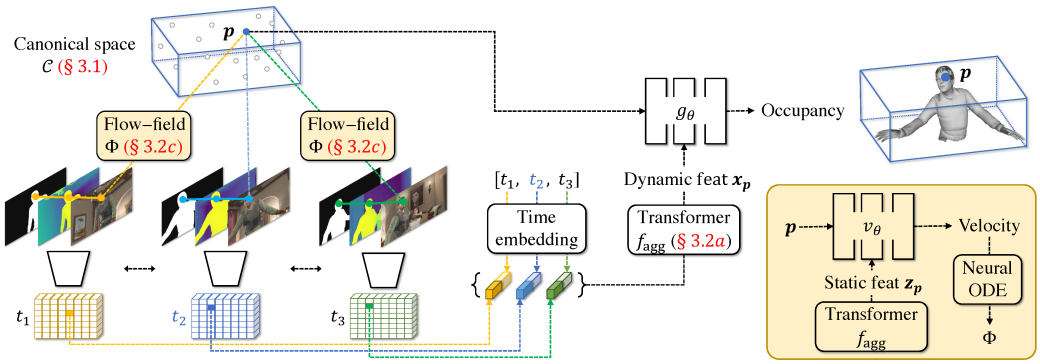


Figure 2: **Framework overview.** For a query point $\mathbf{p}$ in canonical space, REDO first computes the pixel-aligned features from the feature maps of different frames using the flow-field Φ. It then aggregates these features using the temporal aggregator f_{agg}. The obtained dynamic feature $\mathbf{x_p}$ is eventually used to compute the occupancy score for shape reconstruction.

practice, discretized representations like a mesh can be easily extracted from implicit functions via algorithms like Marching Cubes [49].

Fusion-based 4D reconstruction. DynamicFusion [57] is the pioneering work for reconstructing non-rigid scenes from depth videos in real-time. It fuses multiple depth frames into a canonical frame to recover the observable surface and adopt a dense volumetric 6D motion field for modeling motion. Another early work [14] reconstructs dynamic objects without explicit motion fields. Improvements of DynamicFusion leverage more RGB information [28], introduce extra regularization [74, 75], develop efficient representations [20], and predict additional surface properties [24]. Importantly, different from the proposed direction, these works only recover the observable surfaces rather then the complete shape. Moreover, these works often fail to handle fast motion and changing occlusion. Note, these models often have no trainable parameters and are more geometry-based.

Learning-based 4D reconstruction. Reconstruction of the complete shape of dynamic objects is considered hard due to big intra-class variations. For popular object categories such as humans or certain animals, supervised learning of object template parameters has been utilized to ease reconstruction [36, 94, 47, 96, 97]. These templates are carefully designed parametric 3D models, which restrict learning to a low-dimensional solution space. However, the expressiveness of templates is limited. For instance, the SMPL [47] human template struggles to capture the clothing or hair styles. In addition, systems relying on templates are class-specific as different object categories need to be parameterized differently. We think it remains illusive to construct a template for every object category. To overcome these issues, OFlow [59] directly learns 4D reconstruction from scratch. However, it predicts articulation in normalized model space and thus overlooks rigid-motion of the object. It also struggles to handle occlusion. When ground-truth 3D models are not available, dynamic neural radiance fields (NeRF) [54, 43, 63, 61] learn an implicit scene representation from videos, but are often scene-specific and don't scale to a class-agnostic setting. Self-supervised methods [91, 35, 41] are promising and learn 4D reconstruction via 2D supervision of differentiable rendering [37, 48]. In contrast, in this work, we present a class-agnostic and template-free framework which learns to recover the shape and dynamics from input videos.

Motion/Performance capture. When restricting ourselves to human modeling, 4D reconstruction is often referred to as motion capture (MoCap). Marker and scanner based MoCap systems [62, 46, 40] are popular in industry, yet inconvenient and expensive. Without expensive sensors, camera-based methods are more applicable but often less competitive. Standard methods rely on a multi-view camera setup and leverage 3D human body models [32, 95, 21, 85]. Active research problems in the MoCap field include template-free methods [16, 15, 11], single-view depth camera methods [2, 92, 93], monocular RGB video methods [25, 79], and fine-grained body parts (*e.g.*, hand, hair, face) methods [72, 50, 89, 33]. In contrast, this paper studies a more generic class-agnostic setting where multiple different non-rigid objects are reconstructed using one unified framework.

Dynamics modeling. In different communities, the task of dynamics modeling is named differently. For 2D dynamics in images, optical flow methods [13, 80, 83] have been widely studied and used. Scene flow methods [43, 45, 31] aim to capture the dynamics of all points in 3D space densely, and are often time-consuming and inefficient. For objects, non-rigid tracking or matching methods [4, 77, 9, 5]

study the dynamics of non-rigid objects. However, these methods often only track the observable surface rather then the complete shape space. Recently, neural implicit functions [43, 59] have been applied to estimate 3D dynamics, which we adapt and further improve through conditioning on pixel-aligned temporally-aggregated visual features.

4D reconstruction datasets. To support 4D reconstruction, a dataset needs to have both ground-truth 3D models and accurate temporal correspondences. Collecting such a dataset in a real-world setting is extremely challenging as it requires either expensive sensors or restricted experimental settings. To simplify the setting, existing data is either class-specific (*e.g.*, human) [29, 87, 3], or of extremely small scale [74, 28], or lacking ground-truth 3D models [5]. Given the progress in computer graphics, synthetic data is becoming increasingly photo-realistic and readily available. DeformingThings4D [42] provides a large collections of humanoid and animal 4D models, but lacks textures and background. SAIL-VOS 3D [27] contains photo-realistic game videos together with various ground-truth. For more details, please see the comparison table in Appendix § B.

3 REDO

We aim to recover the 3D geometry of a dynamic object over space and time given a RGBD video together with instance masks and camera matrices. To achieve this we develop REDO which is illustrated in Fig. 2. Specifically, REDO reconstructs an object shape and its dynamics in a canonical space, which we detail in § 3.1. For this REDO employs a temporal feature aggregator, a pixel-aligned visual feature extractor, and a flow-field, all of which we discuss in § 3.2. These three modules help align dynamics of the object across time and resolve occlusions to condense the most useful information into the canonical space. Lastly, we detail inference (§ 3.3) and training (§ 3.4).

Notation. The input of REDO is a fixed-length video clip denoted via $\{I_1, \dots, I_N\}$. It consists of N RGBD frames $I_i \in [0,1]^{4\times W\times H}$ ($i \in \{1, \dots, N\}$) of width W and height H, each recorded at time-step t_i. For each frame I_i, we also assume the camera matrix and instance mask $m_{ij} \in \{0,1\}^{W\times H}$ are given, where m_{ij} indicates the set of pixels that correspond to object j. For readability, we define the following operations: 1) `propagate`: transforms temporally in 3D; 2) `project`: transforms from 3D to 2D (image space); and 3) `lift`: transforms from 2D (image space) to 3D.

3.1 Canonical 4D implicit function

Dynamic objects deform and move across time and are often occluded or partially out-of-view. In addition, the depth-size ambiguity also challenges reconstruction algorithms. To condense information about the object across both space and time, we leverage a canonical space $\mathcal{C} \subseteq \mathbb{R}^3$, which aims to capture a holistic geometry representation centered around the object of interest. In our case $\mathcal{C}$ denotes a volume-constrained 3-dimensional space. Temporally, the canonical space corresponds to the center frame I_c at time-step t_c with $c = \lceil (1+N)/2 \rceil$.

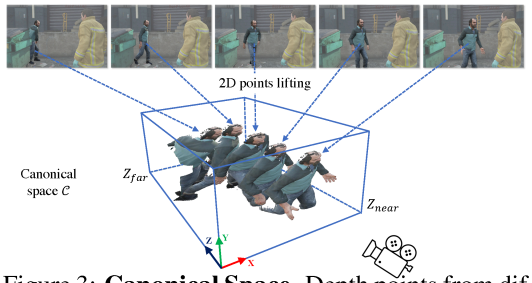


Figure 3: **Canonical Space.** Depth points from different frames are lifted and aggregated to form an enlarged space for capturing the dynamic objects.

To construct the canonical space, we infer a 3D volume around the dynamic object j. For all frames I_i, $i \in \{1, \dots, N\}$, we first `lift` the pixels of instance mask m_{ij} of object j to the world space using the depth map and the camera matrices. These 3D points are then aggregated into canonical space using the camera matrix of the canonical frame. From the aggregated point clouds we infer the horizontal and vertical bounding box coordinates as well as the object's closest distance from the camera, *i.e.*, Z_{near}. However, since the visible pixels only represent the front surface of the observed object, the largest distance of the object from the camera, *i.e.*, Z_{far}, is unknown. To estimate this value, we simply set Z_{far} to stay a fixed distance from Z_{near}. We illustrate an example of this canonical space in Fig. 3, where the inferred canonical space (blue volume) is relatively tight and big enough to capture the complete shape and dynamics of the observed object.

To more precisely capture an object inside the volume-constrained canonical space, we use a 4D implicit function. For each 3D query point in canonical space, this function conditions on a temporal feature to indicate whether that point is inside or outside the object. Specifically, for a query point

inside the canonical space, *i.e.*, for $\mathbf{p} \in \mathcal{C}$, the 4D implicit function is defined as:

$$g_\theta(\mathbf{p}, \mathbf{x_p}) : \mathcal{C} \times \mathbb{R}^K \rightarrow [0, 1]. \tag{1}$$

Intuitively, the higher the score $g_\theta(\cdot, \cdot)$, the more likely the point $\mathbf{p}$ is part of the object. Here, θ subsumes all trainable parameters in the framework and $\mathbf{x_p} \in \mathbb{R}^K$ is a K-dimensional dynamic feature which summarizes information from all input frames. Concretely,

$$\mathbf{x_p} = f_{\text{agg}} \left(\{ f_{\text{enc}}(\psi \left(\Phi(\mathbf{p}, t_c, t_i, v_\theta) \right), I_i) \}_{i=1}^N \right), \tag{2}$$

where $\psi(\cdot) : \mathcal{C} \rightarrow \mathbb{R}^3$ transforms a point in canonical space to world space. Meanwhile, Φ is the flow-field which `propagates` the point $\mathbf{p}$ from t_c to t_i in canonical space, leveraging a learned 3-dimensional velocity-field v_θ. Note, t_c refers to the time-step of the canonical frame I_c while t_i denotes the time-step of frame I_i. Given the transformed point $\psi(\Phi(\mathbf{p}, t_c, t_i, v_\theta)) \in \mathbb{R}^3$, we use a pixel-aligned feature encoder f_{enc} to extract a set of visual representations, each of which comes from one of the N frames I_i, *i.e.*, $\forall i \in \{1, \ldots, N\}$. The set function f_{agg} is a temporal feature aggregator which merges the temporal information of different time-steps. We'll discuss details of temporal aggregator f_{agg}, encoder f_{enc}, flow-field Φ and velocity-field v_θ in § 3.2.

This canonical 4D implicit function `propagates` points across time and extracts the pixel-aligned visual representations from different frames. This differs from: 1) prior works that only consider static objects. Actually, setting $N = 1$ in Eq. (2) will simplify Eq. (1) to $g(\mathbf{p}, f_{\text{enc}}(\psi(\mathbf{p}), I))$, recovering a static Pixel-aligned Implicit Function (PIFu) used in [69, 70]; 2) OFlow [59] which encodes the whole video clip $\{I_1, \ldots, I_N\}$ into one single feature vector, thus loosing spatial information.

3.2 Framework design

In this section, we introduce the temporal aggregator f_{agg}, the feature extractor f_{enc}, and the flow-field Φ used in Eq. (2). These modules help REDO aggregate information from different time-steps (frames) and model the complex dynamics.

a) Temporal aggregator f_{agg}. To deal with partial visibility caused by occlusions or camera settings, we develop a transformer-based temporal aggregator f_{agg} as shown in Fig. 2. f_{agg} is a set function which computes K-dimensional point features $\mathbf{x_p} \in \mathbb{R}^K$. Assume the flow-field Φ is given and object j is of interest. We first `propagate` a query point within the canonical space, *i.e.*, the point $\mathbf{p} \in \mathcal{C}$, to every time-step t_i, obtaining locations $\psi(\Phi(\mathbf{p}, t_c, t_i, v_\theta)) \in \mathbb{R}^3 \; \forall i \in \{1, \ldots, N\}$. We then `project` each 3D point $\psi(\Phi(\mathbf{p}, t_c, t_i, v_\theta))$ back to the corresponding 2D image frame I_i using the associated camera matrix. For all points that are projected into the mask area m_{ij}, we extract a visual representation using the pixel-aligned feature extractor f_{enc}, which we detail below. Note that due to partial visibility, we may not be able to extract features from every frame in the clip. To cope with this, the aggregator f_{agg} is designed as a transformer-based [84] set function. For more information, please see the implementation details in § 4.1 and Appendix § A.

b) Pixel-aligned feature extractor f_{enc}. Pixel-aligned features help REDO make 3D predictions that are aligned with the visual 2D input. To achieve this, we develop $f_{\text{enc}}(\mathbf{q}, I_i)$ where the first argument is a point $\mathbf{q}$ in 3D world space and the second argument is a frame I_i. We first `project` the point $\mathbf{q}$ in world space to the frame I_i, and then use a pre-trained convolutional neural net [58] to extract the 2D feature map of a video frame I_i. For points that fall within the instance mask of a frame, we extract its visual representation using bi-linear interpolation at the `projected` location. We then append a positional encoding [84] of the frame time-step t_i to this feature which helps to retain temporal information. We illustrate this process in Fig. 2. The resulting feature is combined with visual cues from other frames to serve as input to the temporal aggregator f_{agg}.

c) Flow-field Φ. The flow-field models object dynamics in space and time. For this let $\Phi(\mathbf{p}, t_1, t_2, v_\theta) \in \mathcal{C}$ denote a flow-field function in canonical space. It computes the position at time-step t_2 of a 3D point, whose location is $\mathbf{p} \in \mathcal{C}$ at time-step t_1. To compute the displacement, we define a velocity field $v_\theta(\cdot)$ which represents the 3D velocity vectors in space and time via

$$v_\theta(\mathbf{p}, \mathbf{z_p}, t) : \mathcal{C} \times \mathbb{R}^K \times \mathbb{R} \rightarrow \mathcal{C}. \tag{3}$$

Here, $\mathbf{p} \in \mathcal{C}$ is a point in canonical space with corresponding static feature $\mathbf{z_p} \in \mathbb{R}^K$ computed as

$$\mathbf{z_p} = f_{\text{agg}} \left(\{ f_{\text{enc}}(\psi(\mathbf{p}), I_i) \}_{i=1}^N \right). \tag{4}$$

Here, $f_{\text{enc}}(\psi(\mathbf{p}), I_i)$ is the feature of world coordinate point $\psi(\mathbf{p})$ extracted from frame I_i. Note, $\mathbf{z_p}$ differs from $\mathbf{x_p}$ defined in Eq. (2). The feature $\mathbf{z_p}$ summarizes information of *static* locations from all frames I_i. This feature is beneficial as it helps capture whether a point remains static or whether it moves. The velocity network then leverages this feature $\mathbf{z_p}$ to predict object dynamics.

Using the velocity field, we compute the target location at time-step t_2 of a point originating from location $\mathbf{p}$ at time-step t_1 by integrating the velocity field over the interval $[t_1, t_2]$ via

$$\Phi(\mathbf{p}, t_1, t_2, v_\theta) = \mathbf{p} + \int_{t_1}^{t_2} v_\theta(\Phi(\mathbf{p}, t_1, t, v_\theta), \mathbf{z_p}, t)dt. \tag{5}$$

Note that $\Phi(\mathbf{p}, t_1, t_2, v_\theta)$ can represent both forward ($t_2 > t_1$) or backward motion ($t_2 < t_1$) given the initial location and velocity-field. To solve the flow-field for discrete video time-steps, we approximate the above continuous integral equation using a neural-ODE solver [6].

3.3 Inference

To reconstruct objects densely in a clip, the reconstruction/inference procedure is summarized as:

Step 1: We construct the canonical space at the center frame I_c and sample a query point set $\mathcal{P} \subseteq \mathcal{C}$ uniformly from the inferred space. This step requires no network inference and is very efficient.

Step 2: We extract static features $\mathbf{z_p}$ for all $\mathbf{p} \in \mathcal{P}$ using Eq. (4). Next, we compute the trajectory of point $\mathbf{p}$, *i.e.*, $\Phi(\mathbf{p}, t_c, t_i, v_\theta)$ for all time-steps t_i associated with frames I_i, $\forall i \in \{1, \ldots, N\}$, using a neural ODE solver which solves Eq. (5).

Step 3: We then compute dynamic features $\mathbf{x_p}$ for all $\mathbf{p} \in \mathcal{P}$ using Eq. (2).

Step 4: Using Eq. (1) we finally compute the occupancy scores for all $\mathbf{p} \in \mathcal{P}$, which are then transformed into a triangle mesh via Multi-resolution Iso-Surface Extraction (MISE) [53]. Note, the mesh is constructed in canonical space.

Step 5: To obtain the mesh associated with frame I_i, following Step 2, we use the flow-field Φ to `propagate` all vertices of the extracted mesh from the time-step t_c of the canonical space to time-step t_i which corresponds to non-canonical frame I_i. After using the function $\psi(\cdot)$ defined in Eq. (2), we obtain the mesh in 3D world space. While various discrete representations could be obtained from our implicit representation, we use meshes for evaluation and visualization purposes.

3.4 Training

REDO is fully differentiable containing the following parametric components: the temporal aggregator f_{agg}, the feature extractor f_{enc}, the velocity-field network v_θ, and the reconstruction network g_θ. For simplicity, we use θ to subsume all trainable parameters of REDO. To better extract shape and dynamics from given video clips, we train REDO end-to-end using

$$\min_\theta \mathcal{L}_{\text{shape}}(\mathcal{D}, \theta) + \mathcal{L}_{\text{temp}}(\mathcal{D}, \theta), \tag{6}$$

where $\mathcal{D}$ is the training set. $\mathcal{L}_{\text{shape}}(\mathcal{D}, \theta)$ is the shape reconstruction loss in canonical space which encourages REDO to recover the accurate 3D geometry. $\mathcal{L}_{\text{temp}}(\mathcal{D}, \theta)$ is a temporal coherence loss defined on temporal point correspondences. This loss encourages the flow-field Φ to capture the precise dynamics of objects. We detail training set, sampling procedure, and both losses next.

Training set. We train REDO on a dataset $\mathcal{D}$ that consists of entries from different videos and of various object instances. Formally, $\mathcal{D} = \big\{\big(\{(I_i, t_i)\}_{i=1}^N, \mathcal{V}, \mathcal{Y}\big)\big\}$. Specifically, a data entry for an N-frame video clip includes three components: 1) a set of RGBD frames $\{I_i\}$, associated time-steps $\{t_i\}$, as well as corresponding camera matrices $\forall i \in \{1, \ldots, N\}$; 2) an instance ground-truth mesh $\mathcal{V}$ that provides-temporally aligned supervision. For every vertex $\mathbf{v} \in \mathcal{V}$, we use $\mathbf{v}_i$ to denote its position at time-step t_i ; 3) the ground-truth occupancy $\mathcal{Y}$ in the canonical space. Concretely, for a point $\mathbf{p} \in \mathcal{C}$ in canonical space, occupancy label $y(\mathbf{p}) \in \{0, 1\}$ indicates whether $\mathbf{p}$ is inside the object ($y = 1$) or outside the object ($y = 0$). As mentioned in § 3.1, the canonical space is chosen so that it corresponds to time-step t_c, where $c = \lceil (1 + N)/2 \rceil$.

Sampling procedure. To optimize for the parameters θ during training we randomly sample a set of points $\mathbf{p} \in \mathcal{P}(\mathcal{V})$ within the canonical space. $\mathcal{P}(\mathcal{V})$ contains a mixture of uniform sampling and

importance sampling around the ground-truth mesh's surface $\mathcal{V}$ at time-step t_c. Similar strategies are also used in prior works [69, 70].

Shape reconstruction loss. To encourage that the canonical 4D implicit function g_θ accurately captures the shape of objects we use the shape reconstruction loss

$$\mathcal{L}_{\text{shape}}(\mathcal{D}, \theta) = \sum_{\left(\{(I_i, t_i)\}_{i=1}^{N}, \mathcal{V}, \mathcal{Y}\right) \in \mathcal{D}} \frac{1}{|\mathcal{P}(\mathcal{V})|} \sum_{\mathbf{p} \in \mathcal{P}(\mathcal{V})} \text{BCE}\left(g_\theta(\mathbf{p}, \mathbf{x}_\mathbf{p}), y(\mathbf{p})\right), \tag{7}$$

where $\text{BCE}(\cdot, \cdot)$ represents the standard binary cross-entropy loss.

Temporal coherence loss. REDO models dynamics of objects explicitly through the flow-field Φ, which leverages the velocity-field network v_θ. As the ground-truth correspondences across time are available in $\mathcal{V}$, we define the temporal correspondence loss via the squared error

$$\mathcal{L}_{\text{temp}}(\mathcal{D}, \theta) = \sum_{\left(\{(I_i, t_i)\}_{i=1}^{N}, \mathcal{V}, \mathcal{Y}\right) \in \mathcal{D}} \frac{1}{N|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} \sum_{i=1}^{N} \|\Phi(\mathbf{v}_c, t_c, t_i, v_\theta) - \mathbf{v}_i\|_2^2. \tag{8}$$

4 Experiments

We first introduce the key implementation details (§ 4.1) and the experimental setup (§ 4.2), followed by the quantitative results (§ 4.3), qualitative results (§ 4.5), and an in-depth analysis (§ 4.4).

4.1 Implementation details

We briefly introduce the key implementation details. Check Appendix § A for a more detailed version.

Input: We assume all input clips are trimmed to have $N = 17$ frames following [59]. During training, clips are randomly sampled from original videos which have any length. For videos that are shorter than 17 frames, we pad at both ends with duplicated starting and ending frames to form the clips. The validation and test set consist of fixed 17-frame clips. This simplified input setting allows us to split development into manageable pieces, and allows fair comparison with prior work. In practice, dense reconstruction on the entire video is achieved via a sliding window method.

Reconstruction network: Following [69], the reconstruction network g_θ is implemented as a 6-layer MLP with dimensions (259, 1024, 512, 256, 128, 1) and skip-connections. The first layer's dimension of 259 is due to the concatenation of visual features (256-dim) and query point locations (3-dim).

Temporal aggregator: f_{agg} uses a transformer model with 3 multi-headed self-attention blocks and a 1-layer MLP. Group normalization and skip-connections are applied for each block, and we set the hidden dimension to be 128. To compute the time-encoding, we use positional-encoding [84] with 6 exponentially increasing frequencies.

Feature extractor: f_{enc} is implemented as a 2-stack hourglass network [58] following PIFu [69]. Given the instance mask, we take out the object of interest in the picture and resize it to 256×256 before providing it as input to f_{enc}. The output feature map has dimensions of $128 \times 128 \times 256$ of spatial resolution 128×128 and feature dimension $K = 256$.

Velocity-field network: v_θ uses a 4-layer MLP with skip-connections following [59], where the internal dimension is fixed to 128. It takes query points as input and adds the visual features to the activations after the 1^{st} block. For ODE solvers, we use the Dormand–Prince method (dopri5) [12].

Training: In each training iteration we sample 2048 query points for shape reconstruction and 512 vertices for learning of temporal coherence. We train REDO end-to-end using the Adam optimizer [39] for 60 epochs with a batch size of 8. The learning rate is initialized to 0.0001 and decayed by $10\times$ at the 40^{th} and 55^{th} epochs.

4.2 Experimental setup

Dataset. We briefly introduce the three datasets used in our experiments below. For more details (*e.g.*, preparation, statistics, examples), please check Appendix § B.

Dataset	SAIL-VOS 3D			DeformingThings4D++			3DPW		
Metrics	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓
Static reconstruction									
ONet [53]	24.5	0.951	-	**60.2**	**0.260**	-	29.8	0.440	-
PIFuHD [70]	25.6	0.724	-	43.8	0.511	-	37.4	0.363	-
Dynamic reconstruction									
SurfelWarp [20]	1.03	2.13	-	3.75	6.53	-	-	-	-
OFlow [59]	26.0	0.732	1.69	55.2	0.412	0.812	31.5	0.461	0.907
REDO	**31.9**	**0.647**	**1.47**	57.4	0.349	**0.765**	**41.6**	**0.337**	**0.846**

Table 1: **Quantitative results.** For both shape reconstruction (mIoU and mCham.) and dynamics modeling (mACD), REDO demonstrates significant improvements over prior methods. mACD is not available for static methods and SurfelWarp which don't predict temporally corresponding meshes.

1) SAIL-VOS 3D [27]: a photo-realistic synthetic dataset extracted from the game GTA-V. It consists of RGBD videos together with ground-truth (masks and cameras). Out of the original 178 object categories, we use 7 dynamic ones: human, car, truck, motorcycle, bicycle, airplane, and helicopter. During training, we randomly sample clips from 193 training videos. For evaluation, we sample 291 clips from 78 validation videos. We further hold out 2 classes (dog and gorilla) as an unseen test set.

2) DeformingThings4D++: DeformingThings4D [42] is a synthetic dataset containing 39 object categories. As the original dataset only provides texture-less meshes, we render RGBD video and corresponding ground-truth (mask and camera) using Blender. Because the original dataset doesn't provide dataset splits, we create our own. Specifically, during training, we randomly sample clips from 1227 videos. For evaluation, we create a validation set of 152 clips and a test set of 347 clips. We hold out class puma with 56 videos as a zero-shot test set. We dub this dataset DeformingThings4D++.

3) 3D Poses in the Wild (3DPW) [86]: to test the generalizability of our model, we test on this real-world video dataset. Unfortunately, no real-world multi-class 4D dataset is available. Therefore, we test REDO in a class-specific, *i.e.*, class human, setting using 3DPW. This dataset contains calibrated videos, *i.e.*, known camera, and 3D human pose annotation. However, it doesn't provide ground-truth mesh and depth. To extract a mesh, we fit the provided 3D human pose using the SMPL [47] template. To compute depth, we use Consistent Video Depth (CVD) [51] with ground-truth camera data to get temporally consistent estimates. The dataset contains 60 videos (24 training, 12 validation, and 24 testing). During training, we randomly sample clips from all training videos. For evaluation, we evaluate at uniformly sampled clips (10 clips per video) using the validation and test set.

Baselines. We consider the following baselines: 1) Static reconstruction: we adopt state-of-the-art methods ONet [53] and PIFuHD [70], and train them for per-frame static reconstruction. For a fair comparison, we train these two networks in a class-agnostic setting using all the frames in the training videos. 2) Fusion-based dynamic reconstruction: most fusion based methods [57, 74, 75] are neither open-sourced nor reproduced. Among the available ones, we adapt the author-released SurfelWarp [20] due to its superior performance. Since this method is non-parametric and requires no training, we directly apply it on the validation/testing clips. 3) Supervised dynamic reconstruction: REDO learns to reconstruct dynamic objects in a supervised manner. OFlow [59] also falls into this category but it doesn't handle the partial observation and rigid-motion. Note, REDO uses each clip's center frame as the canonical space while OFlow uses the initial one. Therefore, for a fair comparison, we set OFlow's first frame to be the center of our input clip.

Metrics. To evaluate the reconstructed geometry, we report the mean volumetric Intersection over Union (mIoU) and the mean Chamfer ℓ_1 distance (mCham.) over different classes at one time-step, *i.e.*, the center frame of the test clip. To evaluate the temporal motion prediction, we compute the mean Averaged (over time) Correspondence ℓ_2 Distance (mACD) following [59]. As stated before, OFlow's starting frame is set to the center frame. For a fair comparison, we report mACD on the latter half of each testing clip. We compute mCham. and mACD error in a scale-invariant way following [53, 59, 18]: we use 1/10 times the maximal edge length of the object's bounding box as unit one. Even though our network is class-agnostic, we report the mean values over different object categories. Namely, all 'mean' operations are conducted over categories.

4.3 Quantitative results

We present results on all three datasets in Tab. 1. For a fair comparison, we test on the center frame (canonical frame of REDO) of the validation/testing clips. We observe that:

1) REDO improves upon the static methods for shape reconstruction on SAIL-VOS 3D and 3DPW (+6.3/4.2 mIoU and -0.077/-0.026 mCham. over best static method). This is because the static methods cannot capture the visual information from other frames in the video clip and thus fail to handle partial visibility. However, REDO performs slightly worse than ONet on DeformingThings4D++ due to the unrealistic simplified visual input: 1) the pictures have only one foreground object with neither occlusion nor background, 2) the rendered color is determined by the vertices order and hence provides a visual short-cut for 2D to 3D mapping. Without modeling dynamics, the static baselines are hence easier to optimize. In contrast, SAIL-VOS 3D renders photo-realistic game scenes with diverse dynamic objects, which are much closer to a real-world setting.

2) REDO outperforms fusion-based method SurfelWarp greatly on all benchmarks, as SurfelWarp only recovers the observable surface rather than the complete shape. We didn't run SurfelWarp on 3DPW as it relies on precise depth as input and crashes frequently using the estimated depth values.

3) REDO improves upon OFlow (+5.9/2.2/10.1 mIoU and -0.085/0.063/0.124 mCham.) for shape reconstruction due to the pixel-aligned 4D implicit representation, whereas OFlow encodes the whole image as a single feature vector and looses spatial information.

4) Regarding dynamics modeling, REDO improves upon OFlow (-0.22/0.047/0.061 mACD) thanks to the pixel-aligned implicit flow-field. Note that OFlow normalizes the 3D models at each time-step into the first frame's model space and hence fails to capture rigid motion like translation. In contrast, our canonical space is constructed for the entire clip in which REDO predicts a complete trajectory.

As stated in Sec. 3.3, the reconstructed mesh on the center frame is propagated to other frames in the video clip for a dense reconstruction. We thus report the per-frame results in Appendix § E. In addition, all above results are mean values averaged over object categories to avoid being biased towards the most frequent class. Class-wise results on SAIL-VOS 3D are reported in Appendix § C.

Zero-shot reconstruction. To test the generalizability of REDO, we further test on unseen categories with no fine-tuning in Tab. 2. The result is averaged over three unseen classes: dog and gorilla from SAIL-VOS 3D, and puma from DeformingThings4D++. REDO still greatly outperforms baselines and doesn't fail catastrophically. The per-class results are provided in Appendix Tab. S5.

	mIoU↑	mCham.↓	mACD↓
ONet	23.1	0.764	-
PIFu	21.2	0.911	-
SurfelWarp	2.06	1.23	-
OFlow	26.7	0.931	1.18
REDO	**38.5**	**0.479**	**1.07**

Table 2: **Zero-shot reconstruction.**

4.4 Analysis

In Tab. 3, we provide an ablation study of different components in REDO using SAIL-VOS 3D data. 1) We first replace the temporal aggregator f_{agg} with an average pooling layer where features of different frames are averaged and fed into the shape reconstruction and velocity field network. The results are shown in the 1st row of Tab. 3 (avg. pooling). The performance drops by -3.6 IoU, +0.065 mCham., and +0.13 mACD. 2) We then study pixel-aligned feature representations $\mathbf{x_p}, \mathbf{z_p}$. We replace these two features with the feature map of the entire input frame following OFlow [59] but still keep the transformer to aggregate these feature maps. Results of this ablation are reported in Tab. 3 (w/o alignment). Compared to REDO, this setting greatly hurts the results (-7.8 mIoU, +0.290 mCham., -0.38 mACD) as the network can no longer handle partial observations and the 3D predictions don't well align with the visual input. 3) In many real-world tasks, ground-truth meshes of different time-steps are not corresponded. Conceptually, REDO could adapt to this setting. This is because all components are differentiable and the flow-field network could be used as a latent module for shape reconstruction. To mimic this setting, we train REDO using only the shape reconstruction loss $\mathcal{L}_{\text{shape}}$. As shown in Tab. 3 (w/o $\mathcal{L}_{\text{temp}}$), the model still recovers objects at the canonical frame. However, mACD increases significantly (+1.65).

	mIoU↑	mCham.↓	mACD↓
avg. pooling	28.3	0.712	1.60
w/o alignment	24.1	0.937	1.85
w/o $\mathcal{L}_{\text{temp}}$	29.4	0.685	3.12
REDO	**31.9**	**0.647**	**1.47**

Table 3: **Ablation studies.**

4.5 Qualitative results

Fig. 4 shows a few representative examples of REDO predictions on SAIL-VOS 3D and DeformingThings4D++. Please check Appendix § F for more results on real-world data and additional analysis. From Fig. 4 we observe that: 1) REDO is able to recover accurate geometry and dynamics of different objects from input video frames. It completes the occluded parts and hallucinates invisible

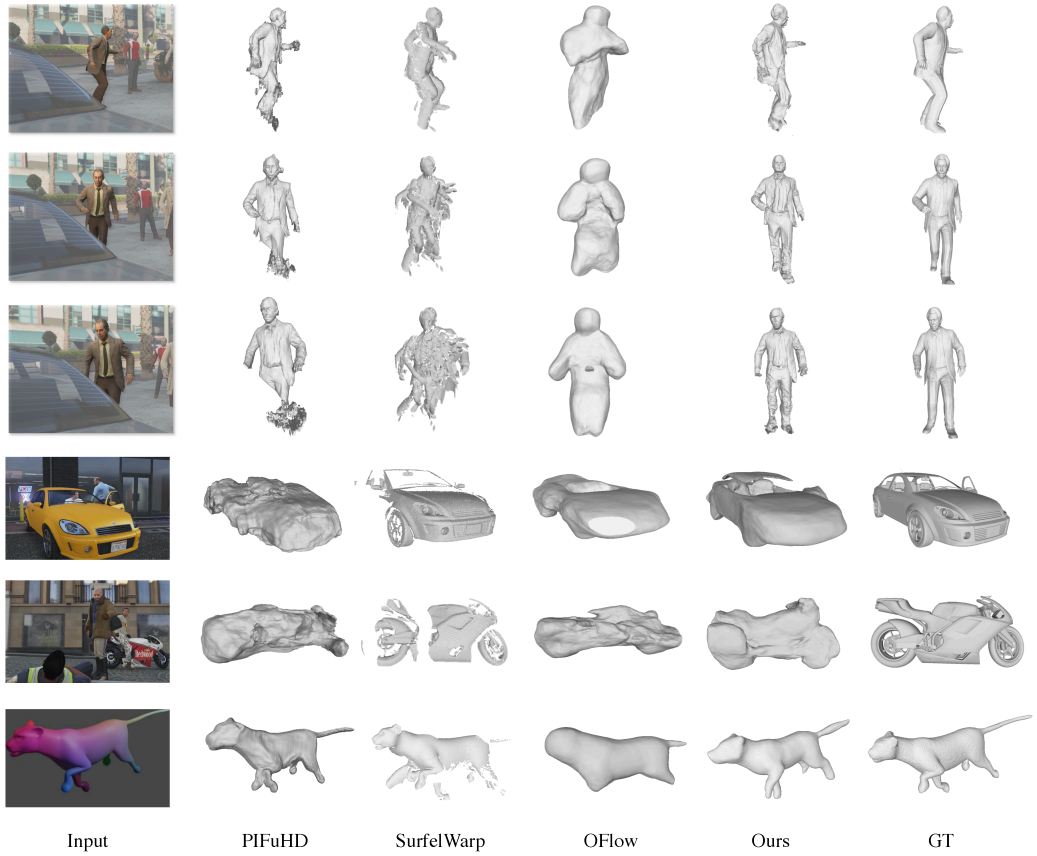


Figure 4: **Qualitative results.** We illustrate the input frames with the object of interest highlighted, the reconstructed meshes obtained from different methods, and the ground-truth mesh.

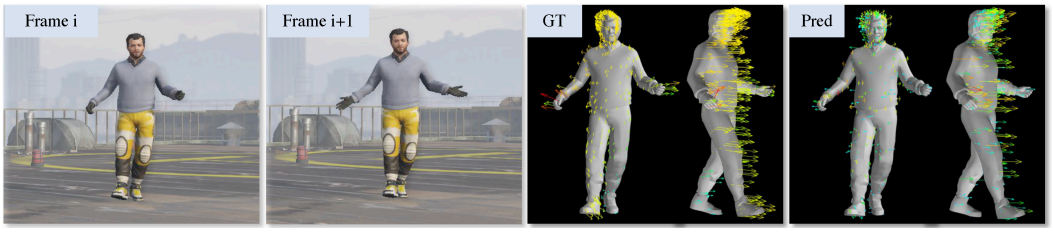


Figure 5: **Flow-field visualization.** REDO accurately recovers the non-rigid motion (*e.g.*, moving forward) but is less precise for small-scale articulation (*e.g.*, hand).

components (*e.g.*, legs and back of humans; rear tire of the motorcycle) by aggregating temporal information and due to large-scale training. 2) REDO improves upon baselines methods. *E.g.*, PIFuHD struggles to handle occlusion and non-human objects, SurfelWarp only predicts the visible surface, and OFlow results are over-smooth as it ignores the spatial information. 3) REDO predictions are still far from perfect compared to ground-truth meshes. Many fine-grained details are missing (*e.g.*, clothing of the human, car's front-light and tires, *etc.*).

We also visualize the predicted and ground-truth motion vectors in Fig. 5. REDO successfully models the rigid motion, *i.e.*, moving forward, over the entire human body, *e.g.*, head, leg, chest, *etc.* It's less accurate in capturing the very fine-grained dynamics, *e.g.*, the hand motion where the ground-truth indicates the hand will open while the prediction doesn't.

5 Conclusion

We present REDO, a novel class-agnostic method to reconstruct dynamic objects from videos. REDO is implemented as a canonical 4D implicit function which captures the precise shape and dynamics and deals with partial visibility. We validate the effectiveness of REDO on synthetic and real-world datasets. We think REDO could be generalized to a wide variety of 4D reconstruction tasks.

Acknowledgements & funding transparency statement. This work was supported in part by NSF under Grant #1718221, 2008387, 2045586, 2106825, MRI #1725729, NIFA award 2020-67021-32799 and Cisco Systems Inc. (Gift Award CG 1377144 - thanks for access to Arcetri). ZR is supported by a Yee Memorial Fund Fellowship.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.

[2] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. *Consumer Depth Cameras for Computer Vision*, 2013.

[3] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, 2017.

[4] A. Bozic, P. Palafox, M. Zollöfer, A. Dai, J. Thies, and M. Nießner. Neural non-rigid tracking. In *NeurIPS*, 2020.

[5] A. Božič, M. Zollhöfer, C. Theobalt, and M. Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *CVPR*, 2020.

[6] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[7] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019.

[8] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016.

[9] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *TOG*, 2015.

[10] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[11] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *TOG*, 2008.

[12] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *J. Comput. Appl. Math*, 1980.

[13] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[14] M. Dou, J. Taylor, H. Fuchs, A. Fitzgibbon, and S. Izadi. 3d scanning deformable objects with a single rgbd sensor. In *CVPR*, 2015.

[15] M. Dou, S. Khamis, Y. Degtyarev, P. L. Davidson, S. Fanello, A. Kowdle, S. Orts, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi. Fusion4d: real-time performance capture of challenging scenes. *TOG*, 2016.

[16] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi. Motion2fusion: Real-time volumetric performance capture. *TOG*, 2017.

[17] Facebook-AI. D2go brings detectron2 to mobile. https://ai.facebook.com/blog/d2go-brings-detectron2-to-mobile/, 2021.

[18] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017.

[19] C. Franklin. Apple unveils new iPad Pro with breakthrough LiDAR Scanner and brings trackpad support to iPadOS. https://www.apple.com/, 2020.

[20] W. Gao and R. Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *RSS*, 2018.

[21] D. Gavrila and L. Davis. Tracking of humans in action: A 3-d model-based approach. *ARPA Image Understanding Workshop*, 1996.

[22] G. Gkioxari, J. Malik, and J. Johnson. Mesh r-cnn. In *ICCV*, 2019.

[23] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018.

[24] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *TOG*, 2017.

[25] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt. Livecap: Real-time human performance capture from monocular video. *TOG*, 2019.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *CVPR*, 2017.

[27] Y.-T. Hu, J. Wang, R. A. Yeh, and A. G. Schwing. SAIL-VOS 3D: A Synthetic Dataset and Baselines for Object Detection and 3D Mesh Reconstruction from Video Data. In *CVPR*, 2021.

[28] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobald, and M. Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *ECCV*, 2016.

[29] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *TPAMI*, 2013.

[30] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. W. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

[31] M. Jaimez, M. Souiai, J. González, and D. Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *ICRA*, 2015.

[32] H. Joo, H. Liu, L. Tan, L. Gui, B. C. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015.

[33] H. Joo, T. Simon, and Y. Sheikh. Total Capture: A 3D Deformation Model for Tracking Faces, Hands, and Bodies. In *CVPR*, 2018.

[34] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end Recovery of Human Shape and Pose. In *CVPR*, 2018.

[35] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning Category-Specific Mesh Reconstruction from Image Collections. In *ECCV*, 2018.

[36] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik. Learning 3d human dynamics from video. In *CVPR*, 2019.

[37] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[38] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Eurographics*, 2006.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[40] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *TOG*, 2009.

[41] X. Li, S. Liu, K. Kim, S. De Mello, V. Jampani, M.-H. Yang, and J. Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *ECCV*, 2020.

[42] Y. Li, H. Takehara, Takafumi, Taketomi, B. Zheng, and M. Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *ICCV*, 2021.

[43] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In *CVPR*, 2021.

[44] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee. Yolactedge: Real-time instance segmentation on the edge. In *ICRA*, 2021.

[45] X. Liu, C. R. Qi, and L. J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, 2019.

[46] M. Loper, N. Mahmood, and M. J. Black. Mosh: Motion and shape capture from sparse markers. *TOG*, 2014.

[47] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *TOG*, 2015.

[48] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *ECCV*, 2014.

[49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *TOG*, 1987.

[50] L. Luo, H. Li, and S. Rusinkiewicz. Structure-aware hair capture. *TOG*, 2013.

[51] X. Luo, J. Huang, R. Szeliski, K. Matzen, and J. Kopf. Consistent video depth estimation. *TOG*, 2020.

[52] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[53] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

[54] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[55] C. Nash, Y. Ganin, S. A. Eslami, and P. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020.

[56] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[57] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015.

[58] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[59] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *ICCV*, 2019.

[60] J. J. Park, P. R. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *CVPR*, 2019.

[61] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021.

[62] S. I. Park and J. K. Hodgins. Capturing and animating skin deformation in human motion. *TOG*, 2006.

[63] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2020.

[64] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016.

[65] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[66] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[67] Z. Ren, I. Misra, A. G. Schwing, and R. Girdhar. 3d spatial recognition without spatially labeled 3d. In *CVPR*, 2021.

[68] G. Riegler, A. Osman Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017.

[69] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization. In *ICCV*, 2019.

[70] S. Saito, T. Simon, J. Saragih, and H. Joo. PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization. In *CVPR*, 2020.

[71] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[72] T. Simon, H. Joo, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[73] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019.

[74] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. KillingFusion: Non-rigid 3D Reconstruction without Correspondences. In *CVPR*, 2017.

[75] M. Slavcheva, M. Baust, and S. Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *CVPR*, 2018.

[76] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *CVPR*, 2016.

[77] J. Starck and A. Hilton. Surface capture for performance-based animation. *CG&A*, 2007.

[78] S. Stein. Lidar on the iPhone 12 Pro. https://www.cnet.com/, 2020.

[79] Z. Su, W. Wan, T. Yu, L. Liu, L. Fang, W. Wang, and Y. Liu. Mulaycap: Multi-layer human performance capture using a monocular video camera. *TVCG*, 2020.

[80] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[81] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *CVPR*, 2021.

[82] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2017.

[83] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[84] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[85] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *TOG*, 2008.

[86] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018.

[87] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018.

[88] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[89] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. *TOG*, 2011.

[90] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016.

[91] G. Yang, D. Sun, V. Jampani, D. Vlasic, F. Cole, H. Chang, D. Ramanan, W. T. Freeman, and C. Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *CVPR*, 2021.

[92] T. Yu, K. Guo, F. Xu, Y. Dong, Z. Su, J. Zhao, J. Li, Q. Dai, and Y. Liu. Bodyfusion: Real-time capture of human motion and surface geometry using a single depth camera. In *ICCV*, 2017.

[93] T. Yu, Z. Zheng, K. Guo, J. Zhao, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *CVPR*, 2018.

[94] J. Y. Zhang, P. Felsen, A. Kanazawa, and J. Malik. Predicting 3d human dynamics from video. In *ICCV*, 2019.

[95] M. Zollhöfer, M. Nießner, S. Izadi, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. W. Fitzgibbon, C. T. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an rgb-d camera. *TOG*, 2014.

[96] S. Zuffi, A. Kanazawa, and M. J. Black. Lions and tigers and bears: Capturing non-rigid 3d articulated shape from images. In *CVPR*, 2018.

[97] S. Zuffi, A. Kanazawa, T. Berger-Wolf, and M. J. Black. Three-D Safari: Learning to Estimate Zebra Pose, Shape, and Texture from Images" In the Wild". In *ICCV*, 2019.

Appendix:
Class-agnostic Reconstruction of Dynamic Objects from Videos

In this appendix we first provide additional implementation details (§ A) before providing more information about the datasets (§ B). We then discuss additional quantitative results including per-class reconstruction results (§ C), the propagated per-frame reconstruction results (§ E), and per-class zero-shot reconstruction results (§ D). Lastly, we illustrate additional qualitative results (§ F).

A Implementation Details

We provide additional implementation details in this section.

Baselines. For the baseline models, we used the released code of ONet [53], OFlow [59] and PIFuHD [69, 70]. For the static 3D reconstruction baselines (*e.g.*, ONet and PIFuHD), we re-train these models on all the frames from the training set using the default setting. Note, the dynamic 4D reconstruction baseline OFlow always utilizes the initial frame for the canonical space. Therefore, for a fair comparison, we set OFlow's first frame to be the center of our input clip.

ODE. To solve the Ordinary Differential Equation (ODE), we use the Dormand–Prince method with a relative error tolerance of $1e^{-2}$ and an absolute error tolerance of $1e^{-4}$. To support batch-wise processing in the ODE solver for the backward flow, we use the variant from OFlow [59]. We also zero-pad the velocity network output to make it compatible with a ODE solver following OFlow [59].

Training. Our code is implemented in PyTorch. The training process takes around 30 hours on a 4 GPU machine. To ensure a fair comparison, parts of our model, *i.e.*, the image encoder and the reconstruction net, share the same architecture as PIFuHD.

B Datasets

We provide a detailed comparison of popular dynamic 3D dataset in Tab. S1. Among them, we choose to use SAIL-VOS 3D, DeformingThings4D, and 3D Poses in the Wild (3DPW) because of the dataset scale and the quality of the ground-truth labels. Besides SAIL-VOS 3D, the other two are incomplete and miss some ground-truth labels. In below section, we introduce the details about how we complete the datasets used in our experiments.

Datasets	Type	#Category	#Videos	RGB	Depth	GT camera	GT mask	GT mesh	Misc. GT
Single-class real-world datasets									
Human3.6M [29]	real-world	1 (human)	N/A	✓	✓	✓	✗	✗	
3DPW [87]	real-world	1 (human)	60	✓	✗	✓	✗	✗	3D pose
D-FAUST [3]	real-world	1 (human)	N/A	✗	✗	✗	✗	✓	
Multi-class real-world datasets									
KillingFusion [74]	real-world	5	5	✓	✓	✓	✓	✗	
VolumeDeform [28]	real-world	8	8	✓	✓	✓	✗	✓	
DeepDeform [5]	real-world	multiple	400	✓	✓	✓	✓	✗	
Multi-class synthetic datasets									
DeformingThings4D++ [42]	synthetic	39	1972	✗	✗	✗	✗	✓	
SAIL-VOS 3D [27]	synthetic	10	484	✓	✓	✓	✓	✓	

Table S1: **Summary of dataset statistics.**

B.1 SAIL-VOS 3D

SAIL-VOS 3D [27] is a very challenging dataset due to the diverse appearance and complex motion. Most observed objects are partially visible and the occlusion is often changing across time. The dataset also has a very challenging long-tail distribution.

Data source. SAIL-VOS 3D is publicly available for research and educational purpose under a user agreement.[2] This dataset is extracted from the photo-realistic game Grand Theft Auto (GTA) V.[3] The collected data of SAIL-VOS 3D contains neither personally identifiable information nor offensive content.

[2] http://sailvos.web.illinois.edu/_site/index.html

[3] https://www.rockstargames.com/games/V

Split	#Videos	#Frames	PO	Size	#Clips	Class-wise distribution
Train	193	475	84%	15%	-	#**classes = 7**: person (188), car (62), truck (10), helicopter (7), motorcycle (10), bicycle (3), airplane (3)
Val	78	675	86%	13%	291	#**classes = 8**: person (61-241), car (12-23), motorcycle (4-8), truck (2-7), bicycle (2-5), trolley (2-3), airplane (1-2), helicopter (1-2)
Unseen	2	333	100%	6.4%	2	#**classes = 2**: dog (1-1), gorilla (1-1)

Table S2: **SAIL-VOD 3D statistics.** Values in parenthesis $(V - C)$ indicate the number of videos V and clips C for the corresponding class. For each split we report the number of videos (#Videos), average video frames (#Frames), partial visibility ratio (PO), ratio between object area and image area (Size), and the number of sampled clips (#Clips). Note, 1) videos are untrimmed and may contain several objects from different classes. Therefore, #Videos may not equal the summation of V from all classes; 2) #Clips for split train is not reported as we randomly sample clips from videos during training.

Splits & statistics. Our network is class-agnostic and handles different dynamic objects in a unified manner. For this purpose, we use 10 kinds of dynamic objects, *i.e.*, human, car, truck, motorcycle, bicycle, airplane, helicopter, gorilla, trolley, and dog. We provide detailed statistics in Tab. S2

Clips sampling. SAIL-VOS 3D contains videos whose number of frames ranges from single digits to several hundreds. For our experiments, we sample fixed-length video clips satisfying the following rules: 1) the average occlusion ratio of the observed object is lower than 0.75 following [27]. 2) the average visible instance area of the object is bigger than 128×128 pixels. In our training, we set the number of frames to be 17 following [59]. If the length of the original video clip is shorter than 17, we duplicate beginning and ending frames to meet this requirement.

Ground-truth. To compute the ground-truth occupancy label, we first truncate the ground-truth mesh of each frame in the input clip to remove the vertices and faces that are never seen in this clip. Note, the set of invisible vertices and faces can be easily identified since SAIL-VOS 3D provides the vertices' temporal correspondences across frames. Therefore, we just need to compute a per-frame set of invisible vertices and faces and then take a union of those per-frame sets. We can then compute the ground-truth occupancy label and correspondences using the converted truncated mesh.

B.2 DeformingThings4D++

Data source. The original dataset DeformingThings4D [42] is released for non-commercial research and educational purposes under "DeformingThings4D Terms of Use"[4], and the accompany code is release under a non-commercial creative commons license. The characters in this dataset are obtained from Adobe Mixamo[5]. This dataset is synthetic and contains neither personally identifiable information nor offensive content.

Rendering. The original DeformingThings4D [42] data is composed of 1972 scenes with 1772 animals and 200 humanoids from 39 classes. Note, the number of classes isn't 31 as listed in the paper [42]. This is because the dataset does not contain a mapping between scene IDs and categories. Therefore, we count classes ourselves and treat each unique object type as a class. See Tab. S3 for more details. Each scene in the dataset contains a triangular mesh and vertices's 3D coordinates at different time steps, which provide ground truth motion fields. For each scene, we render with resolution of 500×600 and obtain the following information with several representative examples shown in Fig. S1.

• Camera: we do not consider camera shot changes and fix the camera pose for the whole animation. Specifically, we first render with a fixed camera pose. If the resulting animations are of low-quality, *e.g.*, too zoomed-in or zoomed-out, we manually adjust to a better pose.

• RGB: in order to maintain correspondences between rendered frames, *i.e.*, same position of an object should have same color across the whole animation, we assign an RGB value to each

[4] https://docs.google.com/forms/d/e/1FAIpQLSckMLPBO8HB8gJsIXFQHtYVQaTPTdd-rZQzyr9LIIkHA515Sg/viewform
[5] https://mixamo.com/

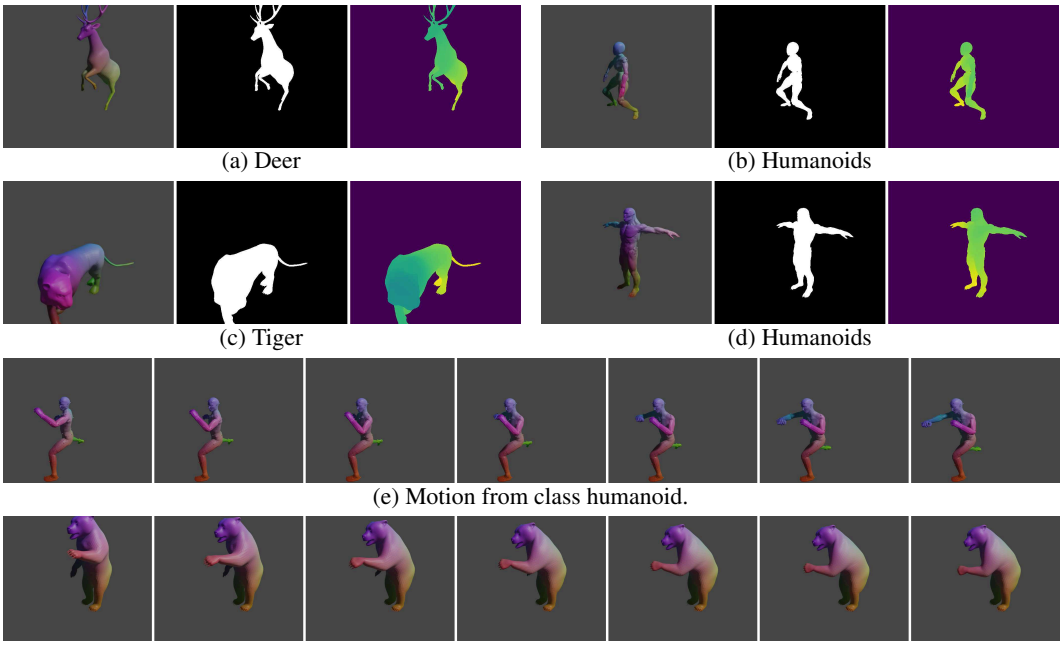
Split	#Videos	#Frames	PO	Size	#Clips	Class-wise distribution
Train	1227	61	88%	8.5%	-	#**classes = 37**: bear (195), deer (170), fox (126), humanoids (111), moose (105), rabbit (72), doggie (45), dragon (41), elk (41), tiger (40), procy (38), raccoon (37), bunny (28), bucks (25), grizz (24), canie (22), huskydog (21), bull (13), milkcow (12), cattle (11), hog (7), rhino (7), cetacea (6), elephant (6), chicken (5), hippo (3), lioness (3), sheep (3), raven (2), cat (1), crocodile (1), duck (1), goat (1), leopard (1), pig (1), seabird (1), zebra (1)
Val	152	64	96%	6.9%	152	#**classes = 25**: bear (27-27), deer (24-24), fox (18-18), moose (15-15), rabbit (10-10), humanoids (7-7), doggie (6-6), dragon (5-5), elk (5-5), procy (5-5), raccoon (5-5), tiger (5-5), bunny (4-4), bucks (3-3), canie (2-2), huskydog (2-2), bull (1-1), cattle (1-1), cetacea (1-1), chicken (1-1), elephant (1-1), hog (1-1), lioness (1-1), milkcow (1-1), sheep (1-1)
Test	347	65	94%	7.2%	347	#**classes = 29**: bear (55-55), deer (48-48), humanoids (44-44), fox (36-36), moose (30-30), rabbit (20-20), doggie (12-12), dragon (11-11), elk (11-11), tiger (11-11), procy (10-10), raccoon (10-10), bunny (8-8), bucks (6-6), grizz (6-6), canie (5-5), huskydog (5-5), bull (3-3), cattle (3-3), milkcow (3-3), hog (2-2), cetacea (1-1), chicken (1-1), duck (1-1), elephant (1-1), goat (1-1), lioness (1-1), raven (1-1), sheep (1-1)
Unseen	56	65	92%	7.4%	56	#**classes = 1**: puma (56-56)

Table S3: **DeformingThings4D++ statistics.** Values in parenthesis $(V - C)$ indicate the number of videos (V) and clips (C) for the corresponding class. For each split we report the number of videos (#Videos), average video frames (#Frames), partial visibility ratio (PO), ratio between object area and image area (Size), and the number of sampled clips (#Clips). Note, #Clips for split train is not reported as we randomly sample clips from videos during training.

vertex on the mesh and keep that RGB value for all time steps. We then utilize Blender's `ShaderNodeVertexColor` to get RGB frames.

• Mask & depth: we ray-cast from each pixel of a frame. 1) For mask, we mark a pixel as "in mask" if its ray reaches the object's mesh. 2) For depth, if the ray reaches the mesh, we store the z coordinate of the intersection point between ray and mesh in the camera coordinate system.

Splits & Statistics. Note, the original dataset doesn't provide the train/validation/test splits. We hence create our own split following the rules discussed next: First, we discard the videos with invalid motion values, *e.g.*, `NaN`. Second, we do not use videos if there are frames where objects leave the view frustum entirely. After filtering, we split the remaining videos into train/validation/test sets with a rough ratio of $7:1:2$: 1) We reserve one class for zero-shot experiments to examine algorithm generalization. Specifically, we remove the class puma from the training/validation/test set as an unseen category. 2) For validation and test sets, we consider a subset of videos that composes of at least one rendered frame where the object is completely visible. The motivation is that if an object appears completely, we do not need to impose any extra processing for evaluation, *e.g.*, truncating the ground truth mesh. In this way, we avoid inaccuracies incurred in the processing step. We sample



(a) Deer (b) Humanoids

(c) Tiger (d) Humanoids

(e) Motion from class humanoid.

(f) Motion from class bear.

Figure S1: **Examples of DeformingThings++. (a) - (d):** we display the rendered RGB image (left), mask (center), and depth (right) of 4 instances from different objects; **(e) - (f):** we showcase dynamics of two objects. Note the rendered images maintain semantic consistency across frames.

152 and 347 videos from the subset for validation and test respectively. 3) For the remaining videos, we use them as the training set. Please check Tab. S3 for detailed split distribution and statistics.

B.3 3D Poses in the Wild (3DPW)

3DPW is a real-world 3D video dataset focusing on humans. The collected videos usually contain multiple people, which are often occluded and out of the camera view. The camera is also moving to capture the human dynamics. Due to all these appealing and challenging properties, we adopt it in our experiments to test the generalizability of REDO under a class-specific setting.

Data source. 3DPW is publicly available under a license.[6] This dataset captures real-world human videos with a moving camera and Inertial Measurement Units (IMUs) in complex scenes. This dataset contains no offensive content.

Ground-truth. For training and evaluation purposes, we estimate the human mesh. Since 3D pose annotation is given, we utilize the Skinned Multi-Person Linear (SMPL) [47] human template. Since the same template is used across all frames in a video, we also get the ground-truth correspondences (vertices). Note that we render an amodal mesh in this experiment under the class-specific setting, meaning we aim to reconstruct the complete human mesh when the visual inputs are partially visible.

C Per-class results

To further analyze the performance of REDO, we provide class-specific results on SAIL-VOS 3D in Tab. S4. Overall, REDO outperforms baseline methods in 18 out of 21 metrics. In terms of performance across classes, REDO performs best for human and car because these two classes have the most training examples. REDO is less effective for airplane and bicycle due to the complex geometry and the small number of training examples.

[6] https://virtualhumans.mpi-inf.mpg.de/3DPW/license.html

Classes	human			car			truck			airplane		
Metrics	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓
ONet [53]	39.2	0.996	-	36.6	0.770	-	24.5	0.942	-	15.7	1.22	-
PIFuHD [70]	42.7	**0.646**	-	28.6	0.726	-	23.2	0.751	-	15.4	0.802	
SurfelWarp [20]	0.841	2.20	-	0.609	3.52	-	0.171	1.72	-	1.54	1.51	
Oflow [59]	37.3	0.748	1.87	35.6	0.858	1.71	21.6	0.801	1.94	21.9	**0.588**	
REDO	**45.2**	0.657	**1.54**	**40.1**	**0.672**	**1.66**	**30.3**	**0.709**	**1.70**	**24.6**	0.615	**0.904**

Classes	helicopter			bicycle			motorcycle		
Metrics	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓
ONet [53]	22.6	0.986	-	13.5	0.898	-	19.3	0.846	-
PIFuHD [70]	25.4	0.725	-	20.4	**0.718**	-	23.3	0.701	-
SurfelWarp [20]	1.42	3.74	-	1.50	5.81	-	1.09	1.10	-
Oflow [59]	24.8	0.526	1.12	18.7	0.831	2.49	22.3	0.770	1.72
REDO	**30.9**	**0.504**	**1.01**	**23.4**	0.730	**1.95**	**28.5**	**0.639**	**1.53**

Table S4: **Per-class reconstruction results on SAIL-VOS 3D.**

D Zero-shot reconstruction

To demonstrate REDO's class-agnostic characteristics and generalization ability, we evaluate REDO on held-out classes. Specifically, we test on classes dog and gorilla from SAIL-VOS 3D and puma from DeformingThings4D++ in Tab. S5. REDO outperforms baselines for unseen classes dog and gorilla on synthetic dataset SAIL-VOS 3D and is competitive on puma from DeformingThings4D++. For class puma, SurfelWarp produces really low IoU while maintaining low Chamfer distance. The reason: although SurfelWarp only reconstructs observable parts as stated in Sec. 4.3, we find the reconstructed geometry overlaps well with the ground truth on this class. Therefore, many vertices on the mesh have near-zero Chamfer distances, resulting in a 0.302 final result in Tab. S5.

Dataset	SAIL-VOS 3D			SAIL-VOS 3D			DeformingThings4D++		
Class	dog			gorilla			puma		
Metrics	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓	mIoU↑	mCham.↓	mACD↓
ONet [53]	23.1	0.727	-	19.5	0.866	-	48.5	0.322	-
PIFuHD [70]	19.3	0.928	-	31.2	0.516	-	46.4	0.389	-
SurfelWarp [20]	0.893	1.46	-	0.984	1.93	-	4.30	0.302	-
Oflow [59]	33.1	0.627	1.51	25.4	0.950	1.37	47.4	0.407	0.661
REDO	36.2	0.597	1.24	34.5	0.403	1.32	44.9	0.436	0.679

Table S5: **Per-class zero-shot reconstruction results.**

E Per-frame results

Recall that REDO reconstructs the mesh in the canonical space and then propagates the reconstructed geometry to every frame in the video clip. To further analyze the performance across frames, we provide dense reconstruction results on SAIL-VOS 3D in Fig. S2. Note, REDO uses each clip's center frame as its canonical space while OFlow uses the 1st frame. Therefore, we show REDO performance for all frames and OFlow for frames from the middle to the last one. As can be seen clearly, REDO outperforms OFlow on all three metrics across frames. However, the performance of both methods drops quickly as the reconstruction is propagated to frames far from the canonical one. This suggests that a better flow-field network is required, which could be an interesting future research direction.

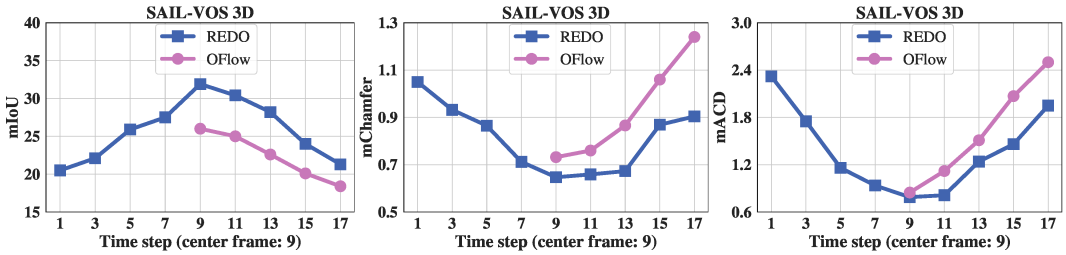


Figure S2: **Per-frame results.** We show mIoU, mCham., mACD from left to right.

F Additional qualitative results

In Fig. S3, we show additional qualitative comparisons on the real-world 3DPW dataset. In each sub-figure we show the input frame, the estimated mask of the object of interest using Mask R-CNN [26], the prediction of the best-performing baseline model (PIFuHD), and our prediction. From Fig. S3 we observe that REDO recovers complete and smooth prediction of the real-world human in various poses. In contrast, the baseline method often struggles when humans are in rare poses. This is largely due to the difference in the pre-training dataset where PIFuHD is trained on stand-still humans with no occlusion, whereas REDO is trained on the large-scale synthetic dataset SAIL-VOS 3D which comes with humans in challenging poses and with sever occlusions.

In Fig. S4, we demonstrate our prediction on SAIL-VOS 3D from multi-views in a single video clip. In each chunk we show the input frames, ground truth mesh, and predicted mesh of one video clip from top to bottom. From Fig. S4 we can see that the front-view of REDO's prediction aligns well with input images. Meanwhile, REDO also hallucinates decently about the side (partially observed) and back (invisible) of the objects. The predicted meshes are relatively smooth across time.

Figure S3: **Qualitative results on 3DPW.** From left to right, we visualize the input image, the estimated mask of the object of interest, the prediction of the best-performing baseline model (PIFuHD), and our prediction.

Figure S4: **Qualitative results.** We illustrate the input frames with the object of interest highlighted, the ground-truth mesh, and the reconstructed mesh (front and side views).