Category-Agnostic Neural Object Rigging

Guangzhao He[1,*,†] Chen Geng [1,*] Shangzhe Wu[1,2] Jiajun Wu[1]

[1]Stanford University [2]University of Cambridge

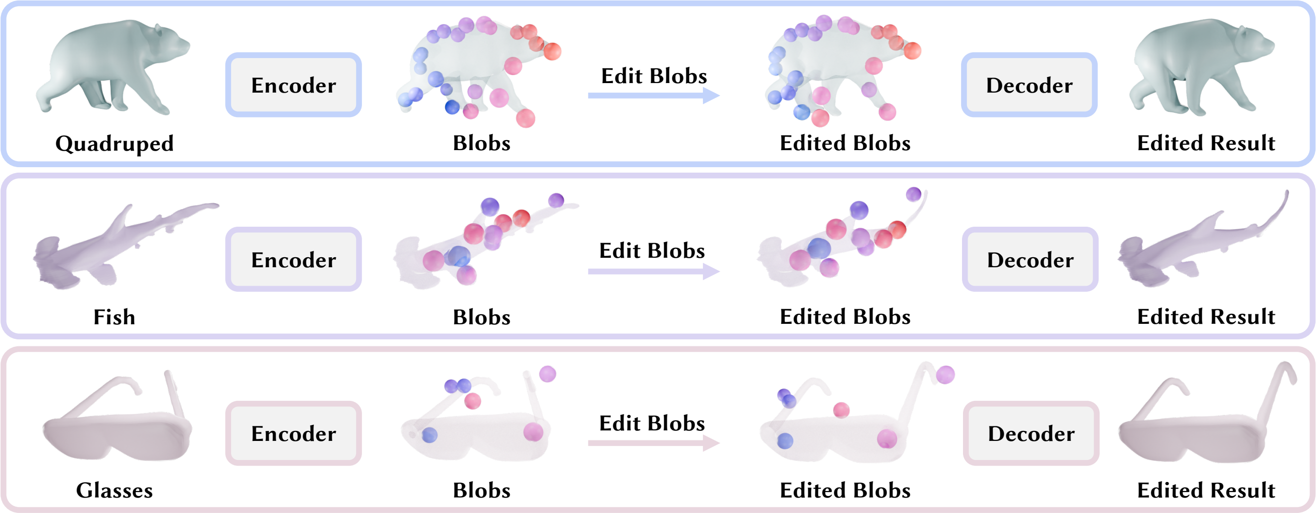

Figure 1. We introduce **Category-Agnostic Neural Object Rigging (CANOR)**, a novel approach that learns to discover a low-dimensional pose space for dynamic objects. The representation is learned from animated 3D sequences of a deformable object category in an unsupervised fashion without relying on any category-specific expert knowledge. By decomposing each object's geometry into a sparse set of feature-embedded blobs, CANOR enables intuitive manipulation of object poses by editing the blobs. This representation captures interpretable motion structures for a diverse range of dynamic object categories.

Abstract

The motion of deformable 4D objects lies in a low-dimensional manifold. To better capture the low dimensionality and enable better controllability, traditional methods have devised several heuristic-based methods, i.e., rigging, for manipulating dynamic objects in an intuitive fashion. However, such representations are not scalable due to the need for expert knowledge of specific categories. Instead, we study the automatic exploration of such low-dimensional structures in a purely data-driven manner. Specifically, we design a novel representation that encodes deformable 4D objects into a sparse set of spatially grounded blobs and an instance-aware feature volume to disentangle the pose and instance information of the 3D shape. With such a representation, we can manipulate the pose of 3D objects intuitively by modifying the parameters of the blobs, while preserving rich instance-specific information. We evaluate the proposed method on a variety of object categories and demonstrate the effectiveness of the proposed framework. Project page: `https://guangzhaohe.com/canor`.

1. Introduction

We live in a dynamic 4D world populated by diverse, ever-moving beings — not just humans, but also pets, wild animals, and other dynamic entities that can move and deform. Modeling and understanding the *structure* of motion across different categories of deformable objects has been a long-standing challenge in Computer Graphics and 3D Computer Vision, with applications in character animation, 4D reconstruction, and AR/VR.

One fundamental property of the motion structure shared by almost all dynamic objects is their inherent low-dimensionality, often captured using various *rigging* representations [31]. Historically, extensive efforts have been

[*]Equal contribution. [†]Work was done when G. He was a visiting student at Stanford University. G. He is currently with Zhejiang University.

dedicated to crafting such representations with domain-specific expertise. For commonly-studied categories, such as humans, domain-specific skeleton structures and skinning methods [5, 19] have been developed. These structured representations significantly facilitate downstream tasks by offering an interpretable motion structure and reliable correspondences across different dynamic states.

While domain-specific representations have been successful for certain dynamic object categories, their development requires extensive expertise, making it impractical to design such structures for every categories of interest. Recently, several methods have attempted to discover similar structures for other dynamic categories without extensive manual intervention [45, 49, 51]. However, most of these approaches still rely on some level of category-specific prior knowledge, which limits their applicability to generic categories.

In this paper, we study the automatic exploration of rigging representations for any dynamic object category, using minimal 3D data, and without any category-specific prior knowledge. Given several animated 3D shape sequences of instances from a certain deformable object category, such as *bears*, we propose an algorithm that extracts the shared pose space within the category. This exploration process is entirely category-agnostic, assuming no prior knowledge of the category, correspondences, or other instance-specific information.

To achieve this, we exploit the key property of rigging representations: their low dimensionality. Drawing inspiration from traditional skeleton-based motion modeling, we introduce a sparse set of spatial-grounded blobs to represent the dynamic poses of moving instances. Given the 3D shape of an instance in a specific pose, we train an encoder to decompose it into dynamic blobs encoding pose information along with instance-aware features capturing instance-specific details. These disentangled bottleneck representations can be further decoded back into the 3D shape.

Once this representation is obtained, we can manipulate the deformable objects in an intuitive manner, where the user can directly drag the extracted blobs to modify object's pose. Moreover, the learned representation reveals a low-dimensional and intuitive structure underlying the high-dimensional deformation space for dynamic objects. To demonstrate these, we apply our method to several diverse object categories that are rarely addressed by prior work, and show significant improvements over state-of-the-art baselines.

In summary, our contributions are:

- We explore a novel task of exploring a category-specific rigging representation in a category-agnostic manner.
- We develop a representation that automatically encodes the 3D shape information into spatially grounded blobs and instance-aware features.
- We evaluate the proposed pipeline on several different deformable object categories and demonstrate significant improvements compared to the State of the Art.

2. Related work

Traditional Rigging Representations. Crafting expressive yet intuitive rigging representations for deformable objects remains a fundamental challenge in the character animation community. For rigging humanoid characters or mammals, the most intuitive method is to annotate their skeletal structures and associated skinning weights [27, 31, 62]. However, this process typically demands substantial manual efforts from artists. Recent works explored on automating the annotation process [3, 46–48]; however, these models often exhibit limited generalization capacities due to insufficient training data. For object categories lacking hierarchical skeleton structures, such as faces, researchers have explored machine learning approaches to develop low-dimensional parametric representations from large databases of aligned shapes [2, 4, 10, 36]. Nevertheless, these approaches typically require substantial amount of high-quality training data, making them challenging to scale for the generic categories considered in this work.

Neural Rigging Representations. Beyond rigging representations with explicit analytical decoding processes, recent research has extensively explored neural-based rigging representations. These approaches leverage deep neural networks to decode latent pose representations into detailed shapes. Our work falls into this category. Within this paradigm, keypoints or handles have emerged as a common control modality [18, 25, 26, 56, 57], which are conceptually similar to the blobs utilized in this work. However, prior works have primarily focused on predicting the deformation of static shapes, whereas our method directly generates posed shapes and targets dynamic objects. Neural Deformation Graphs [7] optimizes node-based representation for rigging dynamic objects, but require a sequence of 3D SDFs as input and lack a learned categorical prior. In contrast, our method performs amortized inference to predict rigging representation directly from a static 3D shape. Another line of work represent complex shapes using learned latent codes without explicit spatial locations [14, 33, 34, 42, 54, 61]. While effective, these representations typically lack interpretability, while our representation encodes the spatial layout of the posed shape in a more intuitive manner.

4D Representations. Prior work on 4D representations often employ low-dimensional structures to regulate motion, including part-based [50, 52, 53], skeleton-based [12, 21, 35, 44, 45], phase-based [13], and node-based [7].

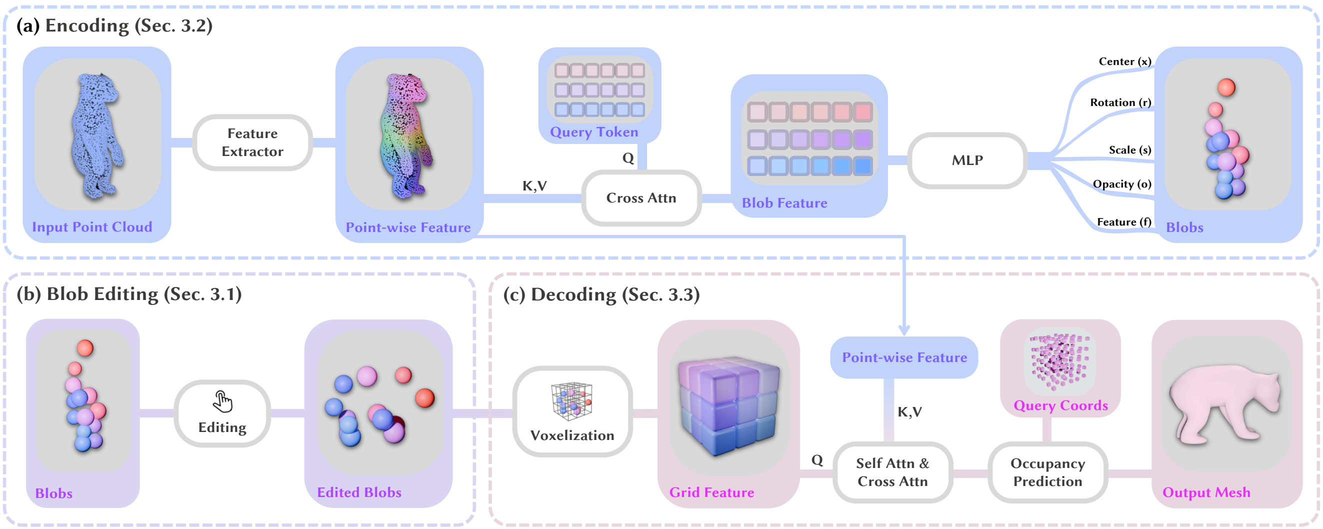

Figure 2. **Overview of our proposed pipeline.** We use a set of feature-embeded blobs to represent the pose space of deformable objects (Sec. 3.1). The encoder takes a point cloud as input and maps it into blobs using a learnable codebook of query tokens that cross-attend with semantic point-wise features (Sec. 3.2). Once generated, these blobs can be edited by users to adjust the object's pose. The edited blobs are then voxelized into a feature volume and decoded back to a 3D shape using a transformer architecture (Sec. 3.3). Finally, the system query the decoded volume with sampled 3D coordinates to predict occupancy values, which are used to extract the edited mesh.

While most of these works rely on rule-based scheme to decode latent into posed shapes, our approach learns a neural decoder directly from data. Other 4D representations [22, 30, 39, 41] directly model high-dimensional flow between different posed shapes. However, these methods often require dense inputs [30] or rely on manually defined regularizations [15, 16, 55].

Mid-level Neural Representations. Our work is also inspired by the large body of research that leverages mid-level neural representations to model visual contents. Most existing approaches focus on capturing scene-level object layouts [8, 11]. Similar to our method, BlobGAN [11] employs blobs as a mid-level neural representation; however, their work is limited to 2D images of indoor scenes. Deep Latent Particles [9] also operates in the 2D domain, demonstrating applications in manipulating human faces using these representations.

3. Method

Given an instance shape from a deformable object category in the form of a point cloud $\mathbf{P} \in \mathbb{R}^{n_p \times 3}$, such as *bears*, *fish*, or *laptops*, our goal is to predict a structured and interpretable representation $\mathcal{B}$ that captures the dynamic *poses* of the object. This representation can be intuitively edited by users to animate or re-pose the 3D object. Inspired by the concept of skeleton-based rigging representations [27, 31], we introduce a sparse set of *blobs* to implicitly encode the spatial structure of dynamic objects. Each blob $\mathbf{b}_i \in \mathcal{B}$ is an anisotropic sphere parameterized by its position, rotation, radius, and feature, which collectively indicate how a semantic part of a dynamic object is positioned at a certain pose.

We learn to discover such a representation in an unsupervised manner. Given the input point cloud $\mathbf{P}$, we define an encoder $\mathcal{E}(\mathbf{P}) = \mathcal{B}$ that maps $\mathbf{P}$ to a sparse set of blobs representing its current pose. A decoder $\mathcal{D}(\mathcal{B}; \mathbf{P})$ is trained to reconstruct the object shape as a mesh. The position and rotation of each blob can be edited to create a novel dynamic pose $\mathcal{B}'$. This modified pose $\mathcal{B}'$ can be subsequently decoded using $\mathcal{D}(\mathcal{B}'; \mathbf{P})$ to generate the re-posed shape of $\mathbf{P}$, as illustrated in Fig. 2.

The following subsections provide a detailed description of our proposed model. We begin by introducing the design of the blob-based representation (Sec. 3.1). In Sec. 3.2 and Sec. 3.3, we detail the architectures for $\mathcal{E}$ and $\mathcal{D}$, respectively. Finally, the training details are discussed in Sec. 3.4.

3.1. Representing Object Pose as Blobs

In this subsection, we describe our rigging representation in the form of a set of blobs.

Blob Parametrization. Each blob in the set captured both spatial and local semantic information corresponding to a specific part of the dynamic object. A blob $\mathbf{b}$ is defined as a feature-embedded anisotropic sphere:

$$\mathbf{b} = (\mathbf{x}, \mathbf{r}, \mathbf{s}, \mathbf{o}, \mathbf{f}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^3$ denotes the center position of the blob, $\mathbf{r} \in \mathbb{H}$ represents its orientation as a rotation quaternion, $\mathbf{s} \in \mathbb{R}$ is the radius of the sphere, $\mathbf{o} \in [0, 1]$ indicates the blob's opacity indicating the activation level, and $\mathbf{f} \in \mathbb{R}^d$ is a feature vector that encodes local semantic information used for shape decoding. With a set of such blobs $\mathcal{B} = \{\mathbf{b}_i \,|\, i = 1, ..., n_b\}$, we can decompose the representation into pose-dependent parameters $\mathcal{B}_P = \{(\mathbf{x}_i, \mathbf{r}_i) \,|\, i = 1, ..., n_b\}$ and identity-dependent parameters $\mathcal{B}_I = \{(\mathbf{s}_i, \mathbf{o}_i, \mathbf{f}_i) \,|\, i = 1, ..., n_b\}$.

Remarks. Blobs offer a flexible, category-agnostic alternative to traditional skeleton-based representations [31] for modeling object poses. Unlike skeletons, which impose rigid hierarchical structures and often require manual design and tuning for different categories, blobs model objects as collections of semi-rigid parts. This makes them easier to learn and generalize across diverse shapes and motion patterns. Moreover, the blob-based design enables intuitive and flexible pose editing. For instance, users can manipulate blob positions ($\mathbf{x}$) and orientations ($\mathbf{r}$) to adjust the object's pose, or modify radius ($\mathbf{s}$) to resize specific parts.

3.2. From Shape to Blobs

Given an input point cloud $\mathbf{P}$, we define a feed-forward encoding process $\mathcal{E}$ that maps $\mathbf{P}$ into a set of blobs $\mathcal{B}$ as described above.

$\mathcal{E}$ consists of two components: $\mathcal{E}_P(\mathbf{P}) = \mathcal{B}_P$, which predicts pose-related parameters, and $\mathcal{E}_I(\mathbf{P}) = \mathcal{B}_I$, which predicts identity-related parameters. This can be formalized as:

$$\mathcal{E}(\mathbf{P}) = \{(\mathcal{E}_P(\mathbf{P})[i], \mathcal{E}_I(\mathbf{P})[i]) \,|\, i = 1, ..., n_b\}. \quad (2)$$

Both encoding processes $\mathcal{E}_P$ and $\mathcal{E}_I$ begin with a shared feature extractor that computes point-wise features $\mathbf{F} \in \mathbb{R}^{n_p \times d}$ from the input point cloud $\mathbf{P}$.

Next, we perform cross-attention between the point-wise features $\mathbf{F}$ and a learnable codebook $\mathcal{Q} = \{\mathbf{q}_i \,|\, i = 1, ..., n_b\}$, where each code corresponds to a distinct blob. This yields attention weights $\mathbf{W} \in \mathbb{R}^{n_b \times n_p}$ between the codebook tokens and input points. The attention weights are used to compute two sets of aggregated feature vectors $\mathcal{F}_P = \{\mathbf{f}_P^i \,|\, i = 1, ..., n_b\}$ for pose and $\mathcal{F}_I = \{\mathbf{f}_I^i \,|\, i = 1, ..., n_b\}$ for identity.

Finally, the blob parameters $\mathcal{B}_P$ and $\mathcal{B}_I$ are regressed from $\mathcal{F}_P$ and $\mathcal{F}_I$, respectively.

Further details of each component are discussed below.

Feature Extractor. To distinguish different semantic parts of the object, maintain pose consistency, and enable accurate shape reconstruction during decoding, the point-wise features $\mathbf{F}$ must encode both semantic and geometric information.

We adopt PointTransformer [59] as our feature extractor due to its strong performance in capturing consistent, expressive, and discriminative features. This capability allows it to recover high-quality details and effectively handle challenging symmetric structures.

Shape Encoding with a Learned Codebook. We aggregate the above-mentioned point-wise feature into a low-dimensional sparse set by learning a shared codebook $\mathcal{Q}$ and performing cross-attention-based feature aggregation. Each token in the codebook corresponds to a distinct blob that captures a specific semantic part. The codebook is learned jointly with other components and shared across object identities to ensure consistency.

To compute the cross-attention-based feature aggregation, we calculate the attention map $\mathbf{W}$ between each token $\mathbf{q} \in \mathcal{Q}$ and each point in the input. The attention weight $\mathbf{W}[i, j]$ between the i-th code and j-th point is given by:

$$\mathbf{W}[i, j] = \frac{\exp\left(\mathcal{Q}[i] \cdot \mathbf{F}[j]]^T / \sqrt{d}\right)}{\sum_{j=1}^{n_p} \exp\left(\mathcal{Q}[i] \cdot \mathbf{F}[j]^T / \sqrt{d}\right)}. \quad (3)$$

The attention weights $\mathbf{W}$ are used to compute two different sets of feature vectors: $\mathcal{F}_P$ for pose and $\mathcal{F}_I$ for identity.

For the pose-related features $\mathcal{F}_P$, we aggregate the positional encodings $\gamma(\cdot)$ [32] of the point coordinates to capture spatial relationships between blobs:

$$\mathcal{F}_P[i] = \sum\nolimits_{j=1}^{n_p} \mathbf{W}[i, j] \cdot \gamma(\mathbf{P}[j]). \quad (4)$$

To extract identity features $\mathcal{F}_I$, which require detailed geometric information, we concatenate the extracted feature $\mathbf{F}$ and point-wise positional encoding $\gamma(\mathbf{P})$, then pass the fused vector through an MLP ϕ:

$$\mathcal{F}_I[i] = \sum\nolimits_{j=1}^{n_p} \mathbf{W}[i, j] \cdot \phi(\mathbf{F}[j] \oplus \gamma(\mathbf{P})[j]), \quad (5)$$

where $\oplus$ denotes the concatenation operator.

Finally, the aggregated feature vector sets $\mathcal{F}_P$ and $\mathcal{F}_P$ are then passed through a group of five shallow MLPs, each responsible for regressing a specific type of blob parameter.

3.3. From Blobs Back to Shape

After encoding the input shape into a set of blobs $\mathcal{B}$, these blobs can be explicitly edited to obtain $\mathcal{B}'$, representing potential pose changes. To reconstruct the re-posed object corresponding to $\mathcal{B}'$, we first voxelize the blobs into a feature volume $\mathbf{V} \in \mathbb{R}^{(h \times w \times l) \times d}$, where h, w and l are the dimensions of the volume. The feature volume is then iteratively refined through a series of self-attention layers [40], and subsequently decoded into an occupancy field that represents the final 3D shape. During this iterative refinement, the volume is also conditioned on the extracted features $\mathbf{F}$ from the encoder to better preserve object identity. We describe the details below.

Blob Feature Volume. Instead of directly passing blob parameters to the decoder, we employ a differentiable voxelization process to ensure that each blob's parameters $\mathbf{x}$, $\mathbf{r}$, $\mathbf{s}$ and $\mathbf{o}$ have explicit and interpretable effects.

To voxelize the blobs, we first construct a 3D grid of coordinates $\mathbf{G} \in \mathbb{R}^{(h \times w \times l) \times 3}$. The feature at each grid point $\mathbf{g}_i$ is computed as a weighted summation of blob features $\mathbf{f}_j$ w.r.t. weight distribution w_{ij}:

$$\mathbf{F}_G[i] = \frac{\sum_{j=0}^{n_b} w_{ij} \cdot \mathbf{f}_j}{\sum_{j=0}^{n_b} w_{ij} + \epsilon}, \tag{6}$$

where ϵ is a small constant to prevent numerical instability. The weights w_{ij} capture the influence of blob j on grid point i, and are defined as:

$$w_{ij} = \mathbf{o}_j \cdot \exp(-c \cdot (\frac{\mathbf{g}_i - \mathbf{x}_j}{\mathbf{s}_j})(\frac{\mathbf{g}_i - \mathbf{x}_j}{\mathbf{s}_j})^T), \tag{7}$$

where c is a constant that controls the softness of the kernel; we set $c = 1$ in all our experiments.

As a special case, w_{i0} and f_{i0} represents a learnable background weight and feature, respectively, which are shared across all inputs.

Augmenting Blob Features. In practice, using only blob features to construct the feature volume may lead to a loss of fine-grained details in reconstruction due to the compactness of the blob representation. To address this, we propose to augment blob features prior to voxelization, enhancing their expressiveness.

Specifically, we enrich each blob feature $\mathbf{f}_j$ with point-specific embeddings $\mathcal{F}$. The augmented feature $\tilde{\mathbf{f}}_{\mathbf{ij}} \in \mathbb{R}^d$ for the i-th grid $\mathbf{g}_i$ and blob $\mathbf{b}_j$ is computed as:

$$\tilde{\mathbf{f}}_{ij} = \mathbf{f}_j + \mathbf{W}_\theta(\gamma(\frac{\mathbf{g}_i - \mathbf{x}_j}{\mathbf{s}_j} \cdot \mathrm{R}(\mathbf{r}_j))), \tag{8}$$

where $\mathbf{W}_\theta$ denotes an MLP with parameters θ and $\mathrm{R}(\cdot)$ denotes the conversion from a quaternion to a 3×3 rotation matrix.

Substituting $\mathbf{f}_{ij}$ with $\tilde{\mathbf{f}}_{ij}$ in the voxelization process described above, we obtain the augmented feature volume $\tilde{\mathbf{F}}_G$:

$$\tilde{\mathbf{F}}_G[i] = \frac{\sum_{j=0}^{n_b} w_{ij} \cdot \tilde{\mathbf{f}}_{ij}}{\sum_{j=0}^{n_b} w_{ij} + \epsilon}. \tag{9}$$

Feature Decoding. Given the augmented grid features $\tilde{\mathbf{F}}_G$, we follow recent shape auto-encoding methods [58, 60] to iteratively process them through a series of self-attention layers to produce $\mathbf{F}'_G$. We add positional encodings $\gamma(\mathbf{G})$ to the features and treat each as an individual token within the attention mechanism.

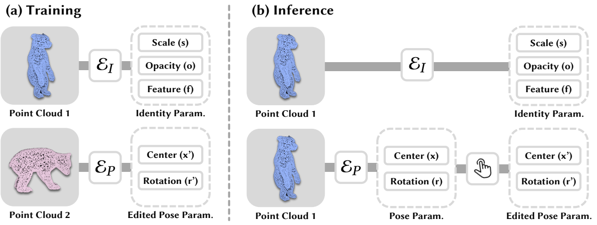

Figure 3. **Difference in training and inference inputs.** During training, we sample two point clouds of the same identity but with different poses to separately predict the identity-related blob parameters $\mathcal{B}_I$ and pose-related parameters $\mathcal{B}_P$. This setup enables $\mathcal{B}_P$ to simulate an edited pose resulting from user edits. During inference, both $\mathcal{B}_I$ and $\mathcal{B}_P$ are predicted from a single point cloud. The user can then explicitly edits $\mathcal{B}_P$ to represent the desired pose change.

To better condition the predicted shape on the identity of the input, we further incorporate cross-attention layers between the grid features and the point-wise features $\mathbf{F}$ extracted during encoding. This conditioning was found to significantly boost the preservation of fine-grained details, as demonstrated in the ablation studies. Addtional details on the decoding architecture are provided in the supplementary material.

Occupancy Prediction. Following prior works [58, 60], for any queried coordinate $\mathbf{x}_q \in \mathbb{R}^3$, our network outputs an occupancy value $o \in [0, 1]$ using the decoded grid features $\mathbf{F}'_G$. We first perform cross-attention between the postional encoding of $\mathbf{x}_q$ and $\mathbf{F}'_G$, and then pass it through an MLP to obtain the final occupancy. We use Marching Cubes [28] to extract the final mesh.

3.4. Training

Training Strategy. The pipeline described above takes an object point cloud $\mathbf{P}$ and user-edited blobs $\mathcal{B}'$ as input and predicts the re-posed mesh $\mathcal{M}'$ of the object. However, directly obtaining the data tuple $(\mathbf{P}, \mathcal{B}', \mathcal{M}')$ is impractical. Instead, we sample two distinct frames of a deformation sequences and obtain their meshes $\mathcal{M}$ and $\mathcal{M}'$ as inputs, ensuring that they share the same object identity but differ in pose. To derive the edited blobs $\mathcal{B}'$, we separately predict pose-related parameters $\mathcal{B}_P$ from $\mathcal{M}'$ and identity-related parameters $\mathcal{B}_I$ from $\mathcal{M}$. Combining these parameters yields blobs that represent the identity of $\mathcal{M}$ with the pose of $\mathcal{M}'$, thus capturing the pose change from $\mathcal{M}$ to $\mathcal{M}'$.

Formally, the training tuple is expressed as:

$$(\mathbf{P}_\mathcal{M}, \{(\mathcal{E}_I(\mathbf{P}_\mathcal{M})[i], \mathcal{E}_P(\mathbf{P}_{\mathcal{M}'})[i])\}, \mathcal{M}'), \tag{10}$$

where $\mathbf{P}_\mathcal{M}$ and $\mathbf{P}'_\mathcal{M}$ denote the sampled point cloud from mesh $\mathcal{M}$ and $\mathcal{M}'$, respectively. The difference between the training and inference input is illustrated in Fig. 3.

Table 1. **Quantitative comparison on DeformingThings4D [24], FaMoS [6], Fish [38], and *Refridgerator* and *Eyeglasses* from articulated objects dataset Shape2Motion [43].** Metrics are averaged over all sequences. IoU, Chamfer Distance L_1 and L_2 are reported. Our method demonstrates significant improvement compared to the state of the arts. Green and yellow cell colors indicate the best and the second best results, respectively.

Method	DeformingThings4D [24]			FaMoS [6]			Fish [38]			Refridgerator			Eyeglasses		
	IoU ↑	CD_1 ↓	CD_2 ↓	IoU ↑	CD_1 ↓	CD_2 ↓	IoU ↑	CD_1 ↓	CD_2 ↓	IoU ↑	CD_1 ↓	CD_2 ↓	IoU ↑	CD_1 ↓	CD_2 ↓
KeypointDeformer [17]	0.536	0.060	0.044	0.923	0.029	0.020	0.499	0.062	0.047	0.744	0.060	0.042	0.452	0.047	0.034
NeuralDeformationGraph [7]	0.875	0.020	0.013	0.800	0.019	0.013	0.686	0.040	0.030	0.869	0.046	0.034	0.791	0.024	0.016
SkeRig [20]	0.802	0.057	0.041	0.790	0.045	0.031	0.782	0.049	0.035	0.803	0.105	0.074	0.544	0.128	0.096
Ours	0.937	0.017	0.011	0.960	0.018	0.013	0.860	0.024	0.017	0.903	0.031	0.022	0.770	0.020	0.014

Training Objectives. We use binary cross entropy loss $\mathcal{L}_{\text{recon}}$ to supervise the proposed model in an end-to-end manner. During training, we apply a sampling mechanism that biases towards near-surface points with a ratio of α_{ns} to enhance high-frequency surface details of the reconstruction. To ensure robust convergence, we additionally regularize the summed grid weights $\mathbf{W}_G = \{\Sigma_{j=1}^{n_b} w_{ij} \,|\, i = 1, ..., h \times w \times l\}$ during voxelization. The regularization term $\mathcal{L}_{\text{vox}}$ is defined as the cosine similarity between $\mathbf{W}_G$ and the GT occupancy of the grid $\mathbf{O}_G$.

The complete training objective is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_{\text{vox}} \mathcal{L}_{\text{vox}}, \tag{11}$$

where λ_{vox} is a hyper-parameter. We refer the readers to the supplementary material for more details on training.

4. Experiments

4.1. Implementation Details

We implement the proposed framework in PyTorch and train it end-to-end using AdamW [29] with a learning rate of $5\mathrm{e}{-4}$. We use 1-Cycle scheduler with linear annealing to accelerate training. We set near-surface sampling ratio α_{ns} to 0.0 for the first 200k iterations to ensure stable gradients for blob initialization, gradually increasing it to 0.5 between 200k and 250k iterations to capture high-frequency details, and finally set it to 0.8 to accelerate convergence. Providing good initializations of blobs before increasing near-surface sampling ratio proved curcial for achieving better spatial distributions of blobs and avoiding local minima. Our training typically converges after 300k iterations, requiring approximately 7 days on 2 RTX A6000 GPUs.

We set the number of blobs [1] n_b to range from 8 to 24 depending on the exact category being modeled. During voxelization, we use a spatial resolution of $8 \times 8 \times 8$, which we find sufficient to capture rich identity details. All attention modules contain 8 self-attention layers implemented using memory-efficient attention [37].

4.2. Evaluating Learned Rigging Representation

Experiment Setup. To evaluate the effectiveness of our learned rigging representation and compare it with the state-of-the-art methods, we assess each method's capability to re-pose a source shape to match a target shape. We then calculate similarity metrics between the re-posed and target shapes. For our method, we directly regress blob positions and orientations from the target shape, using them as pose-related parameters for re-posing. For baselines that do not support feed-forward pose regression, we fix their learned rigging parameters and optimize only pose-related parameters to evaluate their maximum achievable accuracy.

Datasets. We evaluate the proposed method on a diverse set of dynamic object categories to demonstrate its ability to learn rigging representations without relying on category-specific priors. The data includes a quadruped animal dataset DeformingThings4D [24], a human facial expression dataset FaMoS [6], two articulated object categories *Refrigerator* and *Eyeglasses* from Shape2Motion [43], and a custom *Fish* dataset gathered and curated from Sketchfab [38]. Each dataset contains animated sequences of dynamic objects within a single category. For each dataset, we leave out the longest 5 sequences for testing, since some of the baselines methods require hours of optimization for each sequence. We then split the rest of the datasets into training and validation sets to train amortized-inference methods including ours. During testing, we use the first frame from the tested sequence as the source shape and evaluate performance based on the similarity between the re-posed source shape and target shapes from subsequent frames.

Baselines. We compare our method against 3 state-of-the-art baselines with distinct rigging representations. **SkeRig** [20] is a skeleton-based method that optimizes skeleton structures and skinning weights using a small number of frames from a deformable object. It requires additional dense correspondence across frames as input, which may not be available in our training data. Therefore, we estimate per-frame correspondences using a non-rigid registration pipeline NDP [23] and provide them as part of the input to this baseline. To extract the skeleton, we uniformly sample 10 frames from the training set. Once the skeleton is obtained, we freeze its structure and optimize only the bone transformations using differentiable forward kinemat-

[1] We only need to provide an upper bound of the number of blobs.

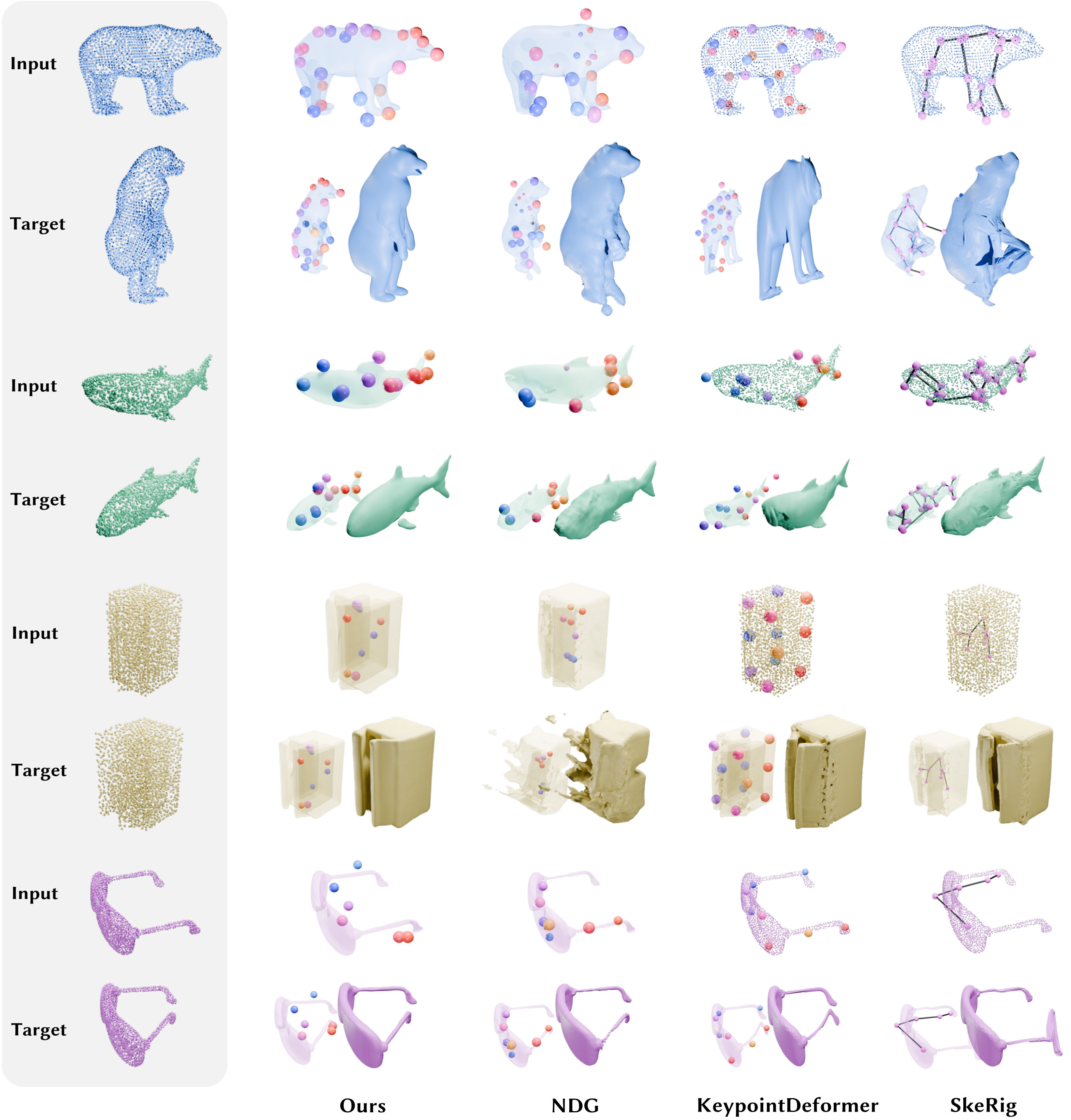

Figure 4. **Qualitative results.** We show qualitative results for different rigging representations across four object categories. Our approach outperforms state-of-the-art methods on both modeling object motion and generating high-quality surface meshes.

ics to best fit the target shapes. **KeypointDeformer** [17] is a keypoint-based method that predicts a set of keypoints for object deformation in a feed-forward manner. It is trained and evaluated using under the same setting as our method. To incorporate more diverse representations, we also implemented an implicit rigging baseline based on **NeuralDeformationGraph** [7], a method that optimizes a neural deformation graph and per-node SDF field for dynamic shape reconstruction. To adapt it to object rigging, we removed its time-dependent implicit shape prediction module. In the first stage of its training, the deformation graph is optimized using frames from the training set. In the second stage, the first frame of the test set is used to optimize the implicit shapes while keeping the graph fixed. After training, we fix the implicit shape and optimize the graph's node positions and rotations to simulate pose editing. We refer the readers to the supplementary materials for additional details.

Metrics. We use three metrics to evaluate the similarity between the target shape and the re-posed source shape: IoU for mesh similarity, and Chamfer L1 and L2 distances for point cloud similarity. Since SkeRig and KeypointDeformer do not estimate surface meshes from input point clouds, we estimate the mesh surfaces for the *refridgerator* and *eyeglasses* datasets — both of which lack ground-truth meshes — using ground-truth occupancy values and Marching Cubes [28] to compute IoU.

Results. We report quantitative comparison results for the five longest sequences from each of the five datasets in Tab. 1. The results show that our method outperforms all baselines by a large margin on nearly all datasets. Qualitative comparisons in Fig. 4 demonstrates that our method accurately models object motion for both rigid and nonrigid dynamic object categories, and generates high-quality surfaces across a variety of shapes. NeuralDeformationGraph[7] produces noisy output, particularly for non-rigid objects, as it tends to overfit training shapes and lacks priors for modeling non-rigid deformations. In contrast, our method learns such priors in a data-driven manner, resulting in natural and accurate mesh surfaces after pose editing. KeypointDeformer [17] fails to capture intricate motions involving topological changes, as it deformes source shapes using cages. Our method mitigates this limitation by using a neural-network-based shape decoder. SkeRig [20] struggles to consistently optimize skeletons across different categories, often producing either over-simplified or overly complex skeletons that fail to capture object motion structure accurately. On the contrary, our method automatically discovers blobs across all categories and consistently represents object motion with high accuracy.

4.3. Animating the "Clay-Monster"

We have demonstrated the effectiveness and accuracy of our method in modeling object deformation on synthetic datasets. To further showcase its potentials in real-world applications—particularly for casual users without 3D modeling expertise—we construct a small object category called *"clay-monster"*, consisting of 12 clay figures scanned in 3 to 5 poses each using only an iPhone. The scans are captured using ARKit 6 [1], and each takes approximately one minute to complete. Using this simple scanning pipeline, we are able to train the proposed model for pose manipulation of such artificial *clay-monsters*, as illustrated in Fig. 5. This demonstrates how animatable representations can be easily obtained for object categories that lack standard rigging information using the proposed pipeline.

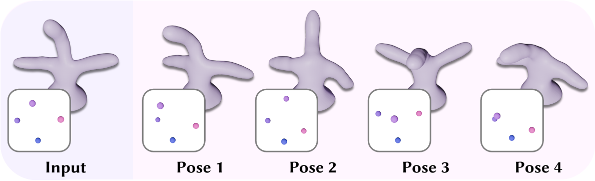

Figure 5. **Pose manipulation results** for a novel category ("clay-monster") using our method, where no rigging tools are available.

Table 2. **Ablation Study.** We show ablation study on two different modules in our method as well as the impact of the number of blobs. All variants are trained on DeformingThings4D [24] dataset. Metrics are averaged over all sequences. IoU, Chamfer Distance L_1 and L_2 are reported.

	IoU ↑	$CD_1\downarrow$	$CD_2\downarrow$
w/o Identity Conditioning	0.853	0.025	0.018
w/o Anisotropic Blobs	0.934	0.017	0.012
Ours $K = 8$	0.845	0.028	0.020
Ours $K = 16$	0.927	0.020	0.013
Ours $K = 24$	0.937	0.017	0.011

4.4. Ablation Study

To validate the design choices of our method, we ablate important components from the proposed pipeline and analyze their impact on performance using the DeformingThings4D [24] dataset. In particular, we ablate the identity conditioning operation and test the use of isotropic blobs to assess their contributions to output quality. We also examine the effect of varying the numbers of blobs, n_b, where the original setting for DeformingThings4D uses $n_b = 24$. As shown in Tab. 2, removing identity conditioning and anisotropic blobs significantly degrades model performance. The results also indicate that increasing the number of blobs improves expressiveness of the model, albeit at the cost of higher computational cost.

5. Conclusion

In this paper, we explored the novel task of learning category-specific rigging representations for dynamic objects in a category-agnostic manner. We introduced a data-driven pipeline for generic object rigging that can be readily applied to any object category. Our method learns to disentangle object pose from identity by representing objects as a set of feature-embedded blobs in a fully unsupervised setting, and reconstruct surface meshes with rich geometric details from these blobs. Experiments across five diverse datasets of distinct object categories demonstrate the effectiveness of our approach.

Acknowledgments. This work is in part supported by ONR YIP N00014-24-1-2117 and NSF RI #2211258 and #2338203.

References

[1] Apple. Arkit 6 - augmented reality. https://developer.apple.com/augmented-reality/arkit, 2024. 8

[2] Stephen W Bailey, Dalton Omens, Paul Dilorenzo, and James F O'Brien. Fast and deep facial deformations. *ACM Transactions on Graphics (TOG)*, 39(4):94–1, 2020. 2

[3] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 2

[4] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 157–164. 2023. 2

[5] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter V. Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. In *Eur. Conf. Comput. Vis.*, pages 561–578, 2016. 2

[6] Timo Bolkart, Tianye Li, and Michael J. Black. Instant multi-view head capture through learnable registration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 768–779, 2023. 6

[7] Aljaz Bozic, Pablo R. Palafox, Michael Zollhöfer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1450–1459, 2021. 2, 6, 7, 8

[8] Chad Carson, Megan Thomas, Serge Belongie, Joseph M Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Visual Information and Information Systems: Third International Conference, VISUAL'99 Amsterdam, The Netherlands, June 2–4, 1999 Proceedings 3*, pages 509–517. Springer, 1999. 3

[9] Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. *arXiv preprint arXiv:2205.15821*, 2022. 3

[10] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (ToG)*, 39(5):1–38, 2020. 2

[11] Dave Epstein, Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A Efros. Blobgan: Spatially disentangled scene representations. In *European conference on computer vision*, pages 616–635. Springer, 2022. 3

[12] Chen Geng, Sida Peng, Zhen Xu, Hujun Bao, and Xiaowei Zhou. Learning neural volumetric representations of dynamic humans in minutes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8770, 2023. 2

[13] Chen Geng, Hong-Xing Yu, Sida Peng, Xiaowei Zhou, and Jiajun Wu. Neural polynomial gabor fields for macro motion analysis. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[14] Simon Giebenhain, Tobias Kirschstein, Markos Georgopoulos, Martin Rünz, Lourdes Agapito, and Matthias Nießner. Learning neural parametric head models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[15] Qixing Huang, Xiangru Huang, Bo Sun, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5815–5825, 2021. 3

[16] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)*, 24(3):1134–1141, 2005. 3

[17] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12783–12792, 2021. 6, 7, 8

[18] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12783–12792, 2021. 2

[19] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, 2007. 2

[20] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph.*, 33(4): 84:1–84:10, 2014. 6, 8

[21] Jiahui Lei, Yufu Wang, Georgios Pavlakos, Lingjie Liu, and Kostas Daniilidis. Gart: Gaussian articulated template models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19876–19887, 2024. 2

[22] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024. 3

[23] Yang Li and Tatsuya Harada. Non-rigid point cloud registration with neural deformation pyramid. In *Adv. Neural Inform. Process. Syst.*, 2022. 6

[24] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Int. Conf. Comput. Vis.*, pages 12686–12696, 2021. 6, 8

[25] Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12–21, 2021. 2

[26] Shaowei Liu, Saurabh Gupta, and Shenlong Wang. Building rearticulable models for arbitrary 3d objects from 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023. 2, 3

[28] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987. 5, 8

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Int. Conf. Learn. Represent.*, 2019. 6

[30] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 3

[31] Nadia Magnenat-Thalmann, Richard Laperrière, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface'88*, pages 26–33, 1989. 1, 2, 3, 4

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Eur. Conf. Comput. Vis.*, pages 405–421, 2020. 4

[33] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. Npms: Neural parametric models for 3d deformable shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12695–12705, 2021. 2

[34] Pablo Palafox, Nikolaos Sarafianos, Tony Tung, and Angela Dai. Spams: Structured implicit parametric models. *CVPR*, 2022. 2

[35] Sida Peng, Chen Geng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Implicit neural representations with structured latent codes for human body modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9895–9907, 2023. 2

[36] Dafei Qin, Jun Saito, Noam Aigerman, Thibault Groueix, and Taku Komura. Neural face rigging for animating and retargeting facial meshes in the wild. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 2

[37] Markus N. Rabe and Charles Staats. Self-attention does not need o(n^2) memory. *CoRR*, 2021. 6

[38] Sketchfab. Sketchfab - the best 3d viewer on the web. https://sketchfab.com, 2024. 6

[39] Colton Stearns, Adam Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic gaussian marbles for novel view synthesis of casual monocular videos. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 3

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, pages 5998–6008, 2017. 4

[41] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024. 3

[42] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1038–1046, 2019. 2

[43] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8876–8884, 2019. 6

[44] Shangzhe Wu, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. DOVE: Learning deformable 3d objects by watching videos. *IJCV*, 2023. 2

[45] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8792–8802, 2023. 2

[46] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 International Conference on 3D Vision (3DV)*, pages 298–307, 2019. 2

[47] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020.

[48] Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. Morig: Motion-aware rigging of character meshes from point clouds. In *SIGGRAPH Asia 2022 Conference Papers*, New York, NY, USA, 2022. Association for Computing Machinery. 2

[49] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 2

[50] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022. 2

[51] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *Advances in Neural Information Processing Systems*, 35:15296–15308, 2022. 2

[52] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *Advances in Neural Information Processing Systems*, 35:15296–15308, 2022. 2

[53] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Hi-lassie: High-fidelity articulated shape and skeleton discovery from sparse image ensemble. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4853–4862, 2023. 2

[54] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 75–83, 2020. 2

[55] Seungwoo Yoo, Kunho Kim, Vladimir G Kim, and Minhyuk Sung. As-plausible-as-possible: Plausibility-aware mesh

deformation using 2d diffusion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4315–4324, 2024. 3

[56] Seungwoo Yoo, Juil Koo, Kyeongmin Yeo, and Minhyuk Sung. Neural pose representation learning for generating and transferring non-rigid object poses. *arXiv preprint arXiv:2406.09728*, 2024. 2

[57] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Trans. Graph.*, 34(4), 2015. 2

[58] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4):92:1–92:16, 2023. 5

[59] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Int. Conf. Comput. Vis.*, pages 16239–16248, 2021. 4

[60] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Adv. Neural Inform. Process. Syst.*, 2023. 5

[61] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 341–357. Springer, 2020. 2

[62] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017. 2