Transfer4D: A framework for frugal motion capture and deformation transfer

Shubh Maheshwari[1] Rahul Narain[2] Ramya Hebbalaguppe[1,2]
[1] TCS Research, India [2] Indian Institute of Technology Delhi, India

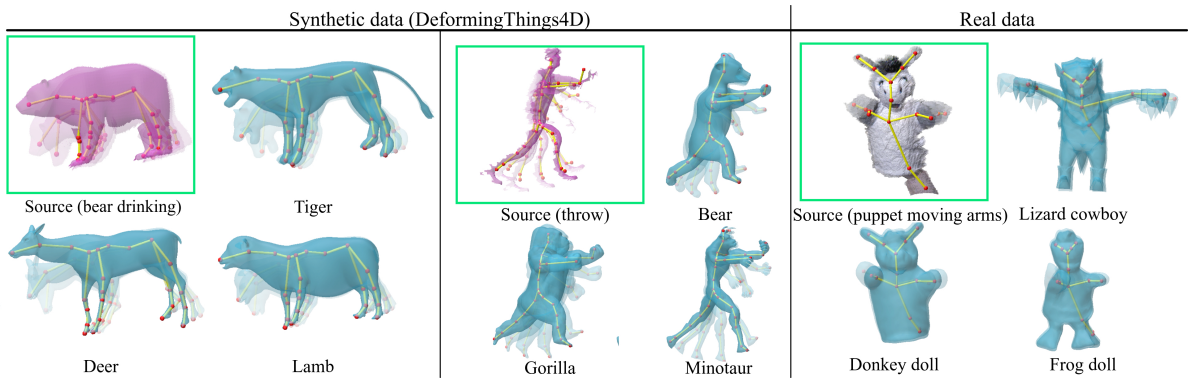

Figure 1. *Transfer4D* is an unsupervised technique for intra-category motion transfer from monocular depth videos (highlighted in the green box) to virtual characters of varying shapes (blue). It is category-agnostic, permitting motion transfer between quadrupeds (left), between bipeds (center and right), and in general between any characters with similar topology. Skeletons shown in yellow are extracted using our proposed unsupervised skeletonization algorithm. Please refer to the demo video on the project webpage: https://transfer4d.github.io/ and Fig. 3 for diverse animation sequences ranging bipeds and quadrupeds.

Abstract

Animating a virtual character based on a real performance of an actor is a challenging task that currently requires expensive motion capture setups and additional effort by expert animators, rendering it accessible only to large production houses. The goal of our work is to democratize this task by developing a frugal alternative termed "Transfer4D" that uses only commodity depth sensors and further reduces animators' effort by automating the rigging and animation transfer process. Our approach can transfer motion from an incomplete, single-view depth video to a semantically similar target mesh, unlike prior works that make a stricter assumption on the source to be noise-free and watertight. To handle sparse, incomplete videos from depth video inputs and variations between source and target objects, we propose to use skeletons as an intermediary representation between motion capture and transfer. We propose a novel unsupervised skeleton extraction pipeline from a single-view depth sequence that incorporates additional geometric information, resulting in superior performance in motion reconstruction and transfer in comparison to the contemporary methods and making our approach generic. We use non-rigid reconstruction to track motion from the depth sequence, and then we rig the source object using skinning decomposition. Finally, the rig is embedded into the target object for motion retargeting.

1. Introduction

The growing demand for immersive animated content originates primarily from its widespread applications in entertainment, metaverse, education, and augmented/virtual reality to name a few. Production of animation content requiring modeling the geometry of the characters of interest, rigging a skeleton to determine each character's degrees of freedom, and then animating their motion. The latter step is often performed by transferring the motion captured from a real performer, whether a human actor, an animal, or even a puppet. The intermediate rigging step is tedious and labor-intensive, while motion capture requires an expensive multi-camera system; both factors hinder the large-scale adoption of 3D content creation. We aim to democratize content creation by (i) replacing expensive motion capture setups with a single monocular depth sensor, and (ii) by automatically generating character rigs to transfer motion to non-isometric objects.

By the term *animation transfer* we refer to the process of transferring the motion captured from a real performer to an articulated character modelled as a polygon mesh. Specif-

ically, we work in a setting where the source is an point cloud obtained from a single-view commodity sensor such as Kinect/Realsense, and the target is a user-defined polygon mesh without a predefined skeletal rig. These choices make our approach more practical for deployment in future. In this setting, performing automatic animation transfer requires reconstructing the source motion from the point cloud, and finding a correspondence between points on the source and target shapes, before the source performance can be remapped to the target character. Furthermore, many applications often require animation content not just of humans but of other kinds of characters and objects such as animals, birds, and articulated mechanical objects, precluding the use of predefined human body models. As we discuss below, existing techniques have limited applicability under all these constraints.

Monocular motion systems generally require a template before registering the incoming video. Templates are created by 360° static registration of the source object [19, 52] or by creating a parametric model from a large corpus of meshes [2, 23, 28, 35, 39, 65]. Unfortunately, these methods do not generalize to unknown objects. As a parallel approach, methods that rely on neural implicit representations [8, 37, 51, 59–61] are comparatively high on computation cost for working on a single video making them unsuitable for frugal animation transfer. Dynamic reconstruction based methods [26, 33, 42, 43, 50] provide promising results, but tracking error could cause structural artifacts thereby resulting in issues of shape correspondence or skeleton extraction.

Without using a predefined template, establishing correspondence between parts of the partial source and the complete target objects becomes challenging. Automatic correspondence methods like [4, 11, 17, 45, 53, 62] do not work in our setting where input is sparse, noisy and incomplete. A family of approaches based on surface level correspondence such as the functional maps [17, 21, 27, 31, 40, 47] incorporate geometry; however, these approaches are data intensive and do not account for the underlying shape topologies explicitly which are critical to matching generic shapes.

Recently, a deep learning approach, Morig [57] was proposed to automatically rig character meshes and capture the motion of performing characters from single-view point cloud streams. However, Morig requires supervision at all stages. Furthermore, Liao et al. [24] introduced a skeleton-free approach to transfer poses between 3D characters. The target object is restricted to bipedal characters in T-pose, and the mocap data is limited to human motion.

To address the above shortcomings, we propose *Transfer4D*, a frugal motion capture and a retargeting framework. We extract the skeleton motion of the source object from a monocular depth video in an unsupervised fashion and retarget the motion to a similar target virtual model. Instead of relying on a predefined template, we directly extract the skeleton from the incoming video. Furthermore, by directly embedding the skeleton into the target object, we bypass establishing dense (surface) correspondence between the source and target that can prove to be computationally expensive and noisy.

Key Contributions: **(1)** To the best of our knowledge, *Transfer4D* is the first approach for intra-category motion transfer technique from a monocular depth video to a target virtual object (Fig. 1). The research aid in efforts towards the democratization of motion-capture systems and reduce animators' drudgery through rigging and automatic motion transfer. **(2)** To transfer motion from a single view incomplete mesh, we propose a novel unsupervised skeleton extraction algorithm. We construct the skeleton based on rigidity constraints from the motion and structural cues from the curve skeleton. See Sec. 4 for details.

2. Related Works

3D Skeleton Extraction: Prior works have studied 3D skeleton extraction as an effective tool for motion transfer [6], shape correspondence [4], retrieval [9], and recognition [12]. Traditionally morphological operations like thinning/erosion [36] or medial axis analysis [48] were used for skeletal extraction. Some data-driven methods like Rignet [55, 56] use neural networks to predict joint location and the skeleton structure. However, RigNet is not effective for skeletonization of incomplete surfaces. Furthermore, RigNet tries to predict a skeleton using only the geometry, whereas our work focuses on computing a skeleton that fits the given motion for transfer. Neural marionette [15] uses Spatio-temporal information to extract key points from a complete mesh sequence. Furthermore, for similar object categories, they can also re-target motion, and generate and interpolate between poses. Unfortunately, these methods do not generalize to single-view incomplete objects. ROSA [49] extracts a curve skeleton from an incomplete point cloud using a rotational symmetry axis. Local Separators [5] transforms mesh to graph structure and used local separators to extract joint. However, these work on static 3D meshes and do not include motion information which is necessary for animation transfer.

To incorporate motion information from single-view point clouds, Zhang et al. [64] propose spectral clustering on the point cloud trajectory to find the joint points for the skeleton. Lu et al. [29] optimize the skeleton extraction procedure in a probabilistic framework using linear blend skinning. Unlike our method these do not utilize curve skeleton information, and consequently, embedding them into the target mesh results in artifacts (See Fig. 4 for additional details).

Real-time Methods to control non-humanoid characters: This line of work focuses on real-time character control by

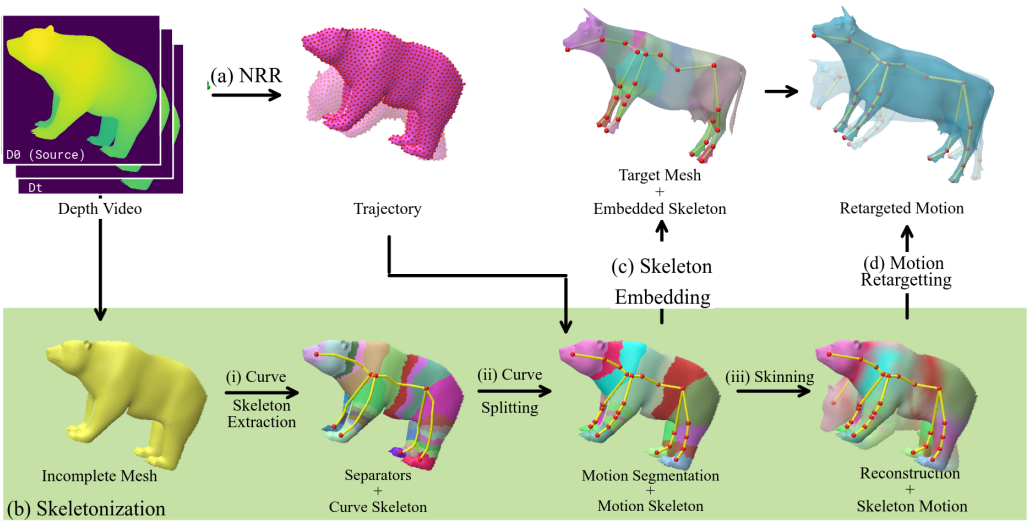

Figure 2. Overview of *Transfer4D* Pipeline. Given a depth video, we first perform *non-rigid registration* (a) to align the source object at a canonical frame to every other frame, yielding its trajectory. Then, *skeletonization* (b) uses the geometry and estimated trajectory of the object to extract its underlying skeletal structure and the motion thereof. In the *skeleton embedding* step (c), the obtained motion skeleton is embedded inside the target mesh and skinning weights are calculated. Finally, for *motion retargetting* (d), the rotation of the bones is transferred from the motion skeleton to the target skeleton. The proposed skeletonization approach, shown in the highlighted green box, takes the incomplete mesh and (i) extracts the curve skeleton and local separator clusters, (ii) splits each curve into bones and the mesh into corresponding motion clusters, and lastly (iii) calculates the skinning weights and transformation parameters of each motion cluster to estimate the skeleton motion.

mapping input from different motion sensors/instruments to poses of animation datasets of non-humanoid characters. Numaguchi et al. [34] create a puppet instrument for motion retrieval. Rhodin et al. [38] estimate wave properties from the skeleton, facial, or hand tracked by off-the-shelf sensors. Anderegg et al. [3] use a smartphone, to tie together to control of the character and camera into a single interaction mechanism. These methods require dedicated sensors for tracking motion and can prove to be expensive. In contrast, our method permits use of a single stationary RGBD camera, without requiring any predefined mapping between the input motion and the target character database.

Skeleton-based motion retargetting: This has been widely studied in the context of humans. Seol et al. [41] and Numaguchi et al. [34] proposed motion puppetry systems to drive the motion of non-human characters. Similarly, Yamane et al. [58] used Kinect-based human skeleton to transfer animation from human to anthropomorphic (human-like eg. bear or monkey) characters. Unlike these approaches, we aim towards generic motion capture where the source and target object are not necessarily humans. Aberman et al. [1] proposed a skeleton-aware deep learning framework, which requires skeletons to be homeomorphic. Hsieh et al. [14] proposed an interactive system to transfer animation between different creatures by manually assigning correspondence between their skeletons. Baran et al. [6] embed the source skeleton to the target shape, calculated attachment weights, and transfer the bone transformation parameters to deform the target shape.

Frugal Motion Capture: To capture motion from a monocular camera is a restricted domain with some representative works: SCAPE [2], SMPL [28] for human bodies, MANO [39] for hands, FLAME [23] for facial expressions, and SMAL [65] for quadrupeds. Similarly, neural-based parametric models like [35, 60, 61] require a large corpus of complete meshes to create. Recent neural implicit-based methods for RBD and RGB-D methods provide a decent alternative to motion-capture-based systems. But their computation cost hinders their utilization for frugal motion capture; at the same time, they do not retarget motion to non-isometric objects.

3. Overview

In our setting, the source is a possibly noisy, single-view depth video with known camera parameters showing a single object in motion. The target is a clean, watertight mesh. We do not have information of part correspondences between the source and target or semantics, making our setting challenging and unsupervised at the same time. Our goal is to transfer the motion of the object in the source video to the target in a visually plausible manner.

Fig. 2 presents a high-level overview of *Transfer4D* pipeline taking an example of incomplete point cloud of

a quadruped (namely, a bear) from a depth camera to retarget motion automatically to a user-defined mesh of a different quadruped (a cow). It comprises of following steps: first, we obtain the source video from the depth camera and segment the object of interest whose motion is to be retargeted; second, we utilize scene flow information and non-rigid registration (NRR) to obtain the motion field of each frame in a scene. Third, we perform skeletonization via our novel approach (Sec. 4) to obtain the position of joints and bones. Finally, the extracted skeleton is embedded into the target user-defined mesh using the Pinnochio framework [6].

Consider a single view depth camera setup which provides a set of depth images, $\mathcal{D} = \{D^t \in \mathbb{R}^{H \times W}\}$, where t, H, W represents timestep, height and width of frames respectively.[1] We segment the object of interest based on the depth values to obtain a binary image mask. We assume the camera is stationary during the recording and camera intrinsic parameters $K \in \mathbb{R}^{3X3}$ are known. Using the camera parameters, for every pixel $u \in \mathbb{R}^2$ in depth image D^t is back-projected to create the point cloud $P = \{p_u\}$.

$$p_u = D^t(u_x, u_y) \cdot K^{-1} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \tag{1}$$

For a depth video of T frames, we choose one frame in the video as the source S. An incomplete mesh $\mathcal{M}_\mathcal{S} = \{V_S, F_S\}$ is obtained from the source depth image where its vertices $V_S = P^S$ and the faces F_S are obtained by connecting adjacent pixels if the distance between their associated vertices is less than a fixed threshold (0.05 in our experiments). Ideally, the source frame should be chosen as the one which best represents the rest state of the object, minimizing noise, self-contact, and self-occlusions. However, for simplicity we always used the first frame of the video as the source frame in our experiments.

3.1. Non-rigid-reconstruction (NRR)

Our first step is to obtain the trajectory of the vertices of the incomplete mesh $\mathcal{M}_\mathcal{S}$ by aligning it to all the future frames using non-rigid registration. To enforce spatial coherence, we represent the motion field using an embedded deformation graph [46] $G = \{V_G, E_G, \mathcal{R}_G^T, \mathcal{T}_G^T\}$. Here each node $g_j \in V_G$ is equipped with a time-varying rigid transformation, i.e. a rotation matrix $R_j^t \in \mathbb{R}^{3\times3}$ and translation vector $\mathcal{T}_j^t$ for each timestep t. The trajectory for each vertex $v_i \in V_S$ at timestep $t \in T$ is computed using the deformation graph as

$$Traj_i^t = \sum_{j=0}^{N_\mathcal{G}} W^\mathcal{G}(i,j)(\mathcal{R}_j^t(v_i - g_j) + g_j + \mathcal{T}_j^t), \tag{2}$$

[1] We experiment using the synthetic datasets processed from DeformingThings4D [63]. The details of preprocessing is described in supplemental material.

where $W^\mathcal{G}(i,j)$ determines the influence of graph node g_j on vertex v_i. It is defined via

$$W^\mathcal{G}(i,j) = \alpha e^{-(\|v_i - g_j\|_2^2)/(2\sigma_{nc}^2)} \tag{3}$$

where α is the normalization constant so that weights sum to one, and the node coverage parameter σ_{nc} controls the weightage of multiple graph nodes on the vertices. We additionally enforce that $\|W^\mathcal{G}(i,:)\|_0 \leq 4$.

We fine-tune the registration using RANSAC and N-ICP [20]. For each timestep t, the graph deformation parameters $\{\mathcal{R}_G^t, \mathcal{T}_G^t\}$, are estimated by optimising the following energy function $E(\mathcal{G})$:

$$\begin{aligned} E(\mathcal{G}) &= \lambda_{corresp} E_{corresp}(\mathcal{G}) + \lambda_{smooth} E_{smooth}(\mathcal{G}), \\ E_{corresp}(\mathcal{G}) &= \sum_{v_i \in \mathcal{V}_\mathcal{S}} \| Traj_i^t - P_x^t \|_2^2, \\ E_{smooth}(\mathcal{G}) &= \sum_{(i,j)\in\mathcal{E}_\mathcal{G}} \|R_i^t(g_j - g_i) + g_i + T_i^t - (g_j + T_j^t)\|_2^2, \end{aligned} \tag{4}$$

where P_x^t is the closest point to $Traj_i^t$ in the new point cloud P^t. In each iteration, K points are sampled from $V_\mathcal{S}$ and transformed using Eq. 2. The data term $E_{corresp}$ seeks to align the incomplete mesh M_S to P^t. The as-rigid-as-possible [44] constraint E_{smooth} enforces nearby graph nodes to have similar transformations. Additional implementation details can be found in the supplementary material.

3.2. Skeletonization

Using the trajectory of the incomplete mesh we want to obtain the motion skeleton $\mathcal{MS}$ to represent the object structure and its motion. To do so, we propose a novel skeletonization technique that utilizes both shape and motion information. We describe this method in full detail in Sec. 4; here we give a brief summary of the approach.

In a nutshell, we first compute a curve skeleton $\mathcal{CS}$ from the mesh geometry and use it to obtain the part segmentation and higher level connectivity of the skeleton (Sec. 4.2). Next, each curve segment and its associated part are subdivided into bones and articulated motion clusters respectively (Sec. 4.3). Lastly, the motion of the skeleton SM is extracted using the articulated motion AM of its vertices (Sec. 4.4).

3.3. Skeleton Embedding

To automate the rigging, we adapt the motion skeleton $\mathcal{MS}$ to the target/reference mesh $\mathcal{M}_T = \{V_T, F_T\}$. Then the adapted target skeleton $\mathcal{TS}$ is attached to $\mathcal{M}_T$ by calculating linear blend skinning weights W_T with similar constraints as Eq. 2. This also alleviates the requirement for the complete surface information for the source shape, unlike dense correspondence techniques like [17].

Finding $\mathcal{TS}$ involves solving maximum subgraph isomorphism between the source skeleton and the medial surface of

the target/reference shape (which can be represented either via a curve skeleton [4] or a graph [6]). The challenge in designing a general shape correspondence algorithm stems from the fact the combinatorial search space is exponential. To solve this difficult problem, we rely on the Pinocchio [6] framework which uses an A* algorithm on all possible matches. In case of large pose and topological variations, Pinocchio may not find matches for all joints.

3.4. Motion Retargetting

We assume the motion that needs to be retargeted should be visible in the video meaning the motion should not be hidden due to self-occlusion. First $\mathcal{MS}$ and $\mathcal{TS}$ are converted to rooted trees. The joint closest to the center of the source mesh, and its corresponding joint in $\mathcal{TS}$, are taken to be the root joints. Parent-child relationships are defined for all other nodes accordingly.

At timestep t, for joint $j \in J_{TS}$ and its parent $p \in J_{TS}$. Rotation matrix $\mathcal{R}_j^t$ aligns the normalized vector $\overrightarrow{j-p}$ to $\overrightarrow{SM_j^t - SM_p^t}$. The motion of the target skeleton TM is computed as:

$$TM_j^t = R_j^t(j-p) + TM_p^t \tag{5}$$

Having specified the motion of the target skeleton, the final re-targetted motion RM of the target mesh M_T can be determined via linear blend skinning:

$$RM_v^t = \sum_{b \in B_{TS}} W(v,b)(R_j^t(v-p) + TM_p^t) \tag{6}$$

where v is the vertex of M_T, and b is the bone in the target skeleton which has joint j and p as its adjacent joints.

4. Proposed Skeletonization

The input to our algorithm is the incomplete mesh $\mathcal{M}_S = \{V_S, F_S\}$ of the source object and its trajectory $Traj \in \mathbb{R}^{T \times |V_S| \times 3}$, where T is the number of timesteps. The trajectory is obtained using Eq. 2 after optimizing the transformation parameters of the deformation graph using NRR. The output is a kinematic pose tree, i.e. a motion skeleton $\mathcal{MS} = \{J_{MS}, B_{MS}\}$ where J_{MS} is a set of joints in $\mathbb{R}^3$ and bones B_{MS} are the edges connecting the joints, along with its skeletal motion $SM \in \mathbb{R}^{T \times |J_{MS}| \times 3}$.

4.1. Preliminaries

We distinguish between a *motion skeleton*, which represents the object's motion degrees of freedom as a hierarchy of nearly rigid parts (i.e. bones) connected by joints, and a *curve skeleton*, which is simply an abstracted 1D representation of an object's geometry via a tree-structured network of curves [13]. Curve-skeletons naturally incorporate the notion of object parts and therefore are used in animation [30], shape retrieval [9] and shape correspondence [4]. In our method, we first compute a curve skeleton to identify the overall topology of the source character, i.e. the head, torso, limbs, etc., and then subdivide the segments of the curve skeleton based on observed motion to obtain the motion skeleton. Our results in Sec. 5 show that the inclusion of a curve skeleton facilitates better correspondence and motion re-targeting.

More precisely, a curve skeleton $\mathcal{CS}$ is a graph, typically a tree, whose nodes are points $j_i \in \mathbb{R}^3$ that together approximate the medial axis of a given shape. We refer to nodes with degree 1, 2, and ≥ 3 as *terminal*, *intermediate*, and *junction nodes* respectively. Terminal and junction nodes together form the *functional nodes*. A path consisting of intermediate nodes between two functional nodes i and k is called a *curve* and denoted C_{ik}.

Additionally, each node i in the curve skeleton is associated with a subset S_i of the vertices of the original shape. These subsets form a partition of the set of shape vertices, i.e. they are disjoint and their union is the entire set V_S. Typically, each subset forms a cylindrical ring around the corresponding skeleton node, or for an incomplete shape, a strip. By taking unions of these subsets, we can associate a *curve region* $\mathcal{CR}_{ik} = \bigcup_{l \in C_{ik}} S_l$ with each curve C_{ik}. These curve regions segment the shape into geometrically significant parts.

4.2. Curve skeleton extraction

We compute the curve skeleton from the incomplete source mesh $\mathcal{M}_S$ using the local separators method of Bærentzen et al. [5]. Their method chooses the skeleton nodes so that each associated S_i is a local separator, i.e. a subset which if removed would disconnect $\mathcal{M}_S$ into two or more disjoint components. Unlike other skeletonization methods, it operates on the graph representation of a mesh, and is not limited to watertight meshes. In the Supplementary, we compare various curve skeleton extraction methods and explain in detail our design choice of using [5].

The curve skeleton produced by their method is not guaranteed to be a tree, so we collapse each cycle into a single node before proceeding. Furthermore, on incomplete meshes resulting from single-view depth images, we find that the algorithm produces spurious short branches and, of course, separate trees for different connected components. To address these issues, we prune the skeleton by removing curves of length ≤ 3, and connect separate components together based on the least edge length variation and boundary information of local separators in pixel space.

Finally, Bærentzen et al.'s method places each node of the curve skeleton at the centroid of its associated mesh vertices S_i. We replace this simple scheme for node placement with an approach inspired from ROSA [49] to better handle incomplete shapes, as follows. We enforce the node to lie near

the medial surface of the object by constraining the node to the intersection of a separator's vertex normals. Hence the optimal position of a node j_i^* is obtained as

$$j_i^* = \arg\min_{j_i \in \mathbb{R}^3} \sum_{v \in S_i} \lambda_{cpp} \|(j_i - v) \times n(v)\|_2^2 + \lambda_{eucl} \|j_i - v\|_2^2 + \lambda_{smooth} \sum_{k \in N(i)} \|j_i - j_k\|_2^2 \quad (7)$$

where $n(v)$ is the vertex normal for vertex v, $N(i)$ are the neighbours of node j_i.

4.3. Curve Partitioning to estimate bones

While the curve regions $\mathcal{CR}_{ik}$ induced by the curve skeleton extraction give us a simple part decomposition of the shape, they do not incorporate any motion information. Therefore, they may need to be subdivided further into functional parts, i.e. rigid bones connected by joints, to obtain the final motion skeleton. We do so by repeatedly splitting each curve C_{ik} to reduce the reconstruction error between the original trajectory and its rigid approximation.

We initialize the motion skeleton $\mathcal{MS}$ by defining a single bone b for each curve C_{ik} in $\mathcal{CS}$. The curve-segment $\mathcal{CR}_{ik}$ represents the motion cluster whose trajectory is governed by the rigid transformation of bone b. Let $\mathcal{R}_b^t, \mathcal{T}_b^t$, be the rotation and translation parameters for bone b at timestep t. We define the reconstruction cost,

$$RC(i,k) = \frac{1}{T} \sum_{t=1}^{T} \sum_{v \in C_{ik}} \|Traj_v^t - (\mathcal{R}_b^t v + \mathcal{T}_b^t)\|_2^2. \quad (8)$$

To find the optimal rotation $\mathcal{R}_b^t$ and translation $\mathcal{T}_b^t$ for bone b we solve the absolute orientation problem [54] using the Kabsch algorithm [16] (explained in Supplementary).

For every bone, we traverse the intermediate nodes of its underlying curve, and find the one at which splitting the curve would give the lowest reconstruction error:

$$r^* = \arg\min_{r \in C_{ik}} RC(i, r-1) + RC(r, k) \quad (9)$$

If the relative change in RC is greater than a threshold ϵ_{split}, we replace the bone with two bones associated with curves C_{ir} and C_{rk} and connected at the joint j_r, and repeat. Otherwise, we keep the original bone.

4.4. Skeleton motion

Lastly, we estimate the deformation of the underlying skeleton which leads to the articulated motion of the source object. This skeleton motion SM will be transferred to the target object during motion retargetting.

To make the bones' transformation deform the surface, the articulated motion $AM \in \mathbb{R}^{T \times |V_S| \times 3}$ is calculated by deforming each vertex v using a weighted combination of bones as defined below,

$$AM_v^t = \sum_{b \in B_{MS}} W(v,b) * (\mathcal{R}_b^t v + \mathcal{T}_b^t) \quad (10)$$

where $W(v,b)$ represents the influence of bone b on vertex v. We compute these skinning weights $W \in \mathbb{R}^{|V_S| \times |B_{MS}|}$ using SSDR [18], which finds the optimal weights subject to the constraints

$$W(v,b) \geq 0, \quad (11)$$

$$\sum_{b \in B_{MS}} W(v,b) = 1, \quad (12)$$

$$|\{b \in B_{MS} : W(v,b) > 0\}| \leq 4. \quad (13)$$

Using the reconstructed motion, the location of each joint at timestep t is obtained as:

$$SM_j^t = j + \sum_{v \in S_j} (AM_v^t - v) \quad (14)$$

5. Experiments and Results

Experimental Setup: We perform the experiments on an Alienware laptop with an Intel i7 CPU, 32GB RAM, and an 8GB Nvidia GeForce GTX 1080. The hyperparameters are listed in the supplementary material for reproducibility.

Datasets and Preprocessing: As our goal is motion transfer between generic real-depth videos to virtual characters of semantically similar but differing shapes, we use DeformingThings4D [63], a dataset of non-rigidly deforming objects. For every example, we computed a source depth video by assigning a random camera view and calculating depth images using Blender Eevee[2]. A detailed description of other benchmark datasets and preprocessing to obtain a single view for our experiments can be found in the supplementary material.

5.1. Comparison with the state-of-the-art

Shape correspondence between dissimilar shapes is an ill-defined problem, and there is no rigorous definition of what constitutes good correspondence of parts from the source to the target. Instead, one can qualitatively identify poor correspondences where semantically different parts are matched to each other, e.g., a mapping of a source character's head to a target character's leg. Conversely, a correspondence that respects the semantics of parts can be considered as a high-quality correspondence. We have tested our approach on a wide variety of model pairs with similar semantic structures (Fig. 1 and Fig. 3 show just a small subset of our results). We find *Transfer4D* to be robust to some degree of variation in poses, although it fails if the character is rotated 180 degrees.

[2]https://docs.blender.org/manual/en/latest/render/eevee/index.html

Method	Percentage ↑ Joints Embedded	Reconstruction ↓ Error	Local ↓ Pose Error
Zhang et al [64]	$31^{\pm 21}$ %	$0.91^{\pm 32}$	$0.36^{\pm 0.18}$
Le et al . [18]	$52^{\pm 38}$ %	$0.71^{\pm 38}$	$0.28^{\pm 0.16}$
Neural Marionette [15]	-	$0.51^{\pm 0.25}$	$0.23^{\pm 0.07}$
Ours	$84^{\pm 22}$ %	$0.37^{\pm 23}$	$0.18^{\pm 0.09}$

Table 1. Overall performance scores of motion retargetting across skeletonization variants

We do not perform any parameter tuning, and use the default thresholds for all results.

Comparison with contemporary methods: We compare our method both qualitatively and quantitatively against a set of well-known unsupervised methods, namely Zhang et al. [64], Le et al. [18], and the recent Neural Marionette [15]. For [64] we use our own implementation since the code is not available. For [18] we use the authors' code for skinning, but perform skeletonization ourselves as the reference code fails on incomplete meshes. For [15] we use only the code provided by the authors. We did not compare against Lu et al. [29] as the source code was not released, and we could not reproduce the results in their manuscript with our own implementation.

In order to have a ground-truth target motion for evaluation, we transfer motion from the incomplete source back to the complete mesh of the same shape as the target. Thus the ideal result is to recover the original motion of the complete mesh. Neural Marionette [15] carries out the entire animation transfer process, while for Zhang et al.and Le et al.we use their method for skeletonization and continue with the remaining steps of our animation transfer pipeline.

Fig. 4 shows a qualitative comparison of motion transfer using our technique and others methods. Tab. 1 shows the quantitative performance scores. We report (i) the percentage of joints that were successfully embedded into the same shape, (ii) the reconstruction error between the re-targeted motion and the original motion, and (iii) the local pose error, i.e. the reconstruction error after rigidly aligning the re-targeted motion with the original motion at each timestep. The supplementary provides details about the performance scores, samples used for evaluation, and additional quantitative comparisons of our technique with state-of-the-art methods.

From Fig. 4 and Tab. 1, it can be observed that our method outperforms all three tested approaches both qualitatively and quantitatively. As Neural Marionette uses a volumetric representation without preprocessing, it is unable to handle an incomplete point cloud sequence. Le et al. [18] and Zhang et al.'s method uses only motion information without structural cues, resulting in a skeleton that is less effective for embedding.

Results on real scanned videos: Fig. 5 shows animation transfer results from real captured RGBD videos. We experimented with three examples. In (a), we transferred the motion of a single-view human puppeteering a doll back to the same object and to biped examples from ModelsResource dataset [56]. In (b) and (c), we used a video of an

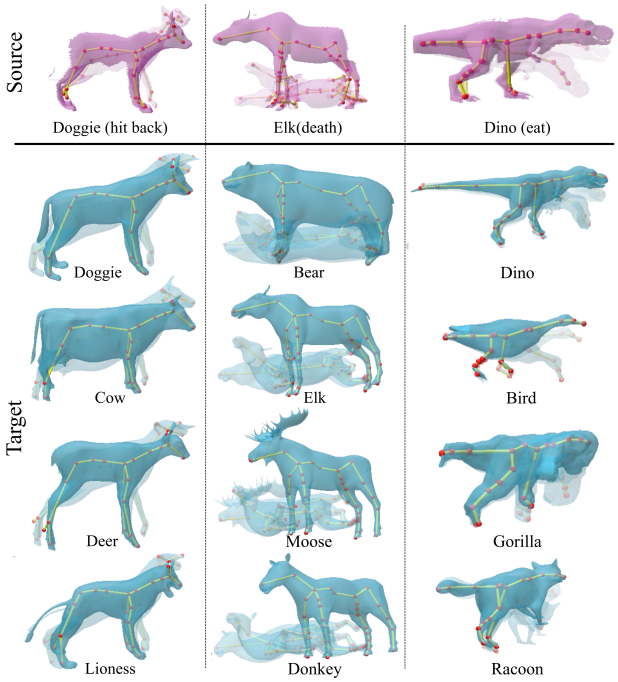

Figure 3. Motion/Animation Transfer results from *Transfer4D* on sequences from DeformingThings4D [63] dataset: Skeleton in the source was extracted using our proposed approach and embedded into the target mesh. Notice that our approach is able to transfer motion to diverse creatures emphasizing the generality of our approach.

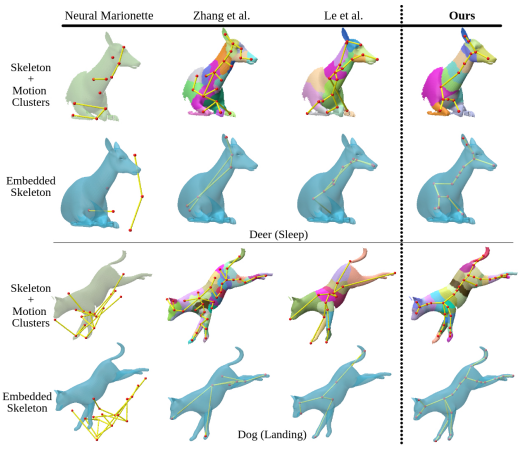

Figure 4. Qualitative comparison of our proposed skeletonization against the baselines: Neural Marionette [15], Zhang et al. [64], Le et al. [18]. **Top row**: the motion skeleton and the motion clusters for each method; note that Neural Marionette only detects key points and does not compute skinning weights. **Bottom row**: the skeleton mapped to the target shape – by Neural Marionette itself, or embedded using Pinocchio [6] in the case of the other methods.

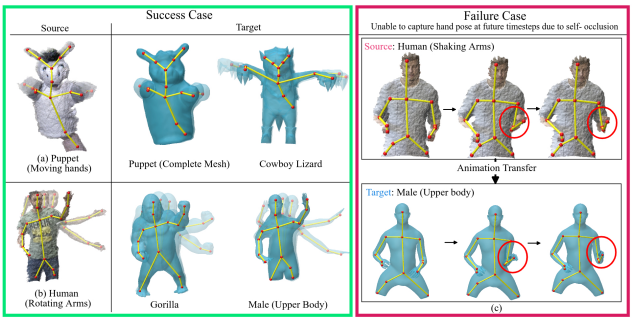

Figure 5. Animation Transfer on real scanned sequences. (a) Donkey doll [52] (b) Adult from DeepDeform [7]. We show successful animation transfer results on the left and a failure case due to self-occlusion on the right.

adult moving his arms from the DeepDeform dataset [7] and retargeted it to a gorilla and upper body of SMPL mesh respectively, with manual editing for plausible motion transfer. However, we encountered a failure case in (c), where self-occlusion prevented us from estimating the pose and retargeting the motion of the left hand, as highlighted in red ellipses. We also observed inaccuracies in embedding the hip joint to other biped examples using Pinocchio.

5.2. Discussion

Our system is aimed to be a frugal alternative to motion capture setups. The supplementary material shows the cost comparison of our setup with the prior works.

With respect to semantic correspondence between source and target shapes: (a) We assume that there would not be large topological variations between the shapes. Since our approach depends on Pinocchio [6] framework for skeletal embedding, if there is a part of the source object which cannot be mapped to the target our approach tends to fail. For example, if the algorithm is used to map a human to an octopus mesh, our approach may not have a corresponding part in the human to match with every part of the octopus. Finding partial embedding could potentially lead to failure in this case. (b) Our approach fails in situations of large pose variations between source and target. In addition, the global orientation of the target shape should be aligned with the source shape (as depicted in Fig. 1). An embedding of a human upside down will also be upside down which leading to an implausible motion transfer.

There are methods such that NeuroMorph [10], and Elector Voters method [4] that target a somewhat similar problem albeit with an assumption of source to be noise-free and complete. Thus, a fair comparison would not be possible with our setting from a single-view frugal sensor which gives rise to incomplete point cloud. Methods like ROSA [49], Point2Skeleton [25] are designed to work on static 3D meshes, and do not incorporate motion information which is necessary for motion transfer.

5.3. Limitations and Future Work

We list a few assumptions and limitations of our method below. **(1)** *NRR:* We use the traditional setup of RANSAC and N-ICP [20] for NRR. This makes our setup susceptible to failure in case of large deformation between the source and target frame. We also conducted the experiments with other methods but found N-ICP performs better on our chosen metrics (see Sec. 2.5 in supplementary) **(2)** *Error propagation:* An error in one stage of our pipeline will affect all subsequent stages. However, we could reduce errors within each step by implementing additional constraints, like temporal smoothing. Furthermore, metrics proposed in Sec 2.5 in supplementary and Tab. 1 can be used to detect a failure in NRR and skeleton embedding, respectively. **(3)** *Detailed motion capture:* Our primary focus is on articulated motion. Fine details such as wrinkles on clothing or facial expressions are not captured or transferred by our system. Likewise, highly nonrigid motions such as loose-fitting dresses or gowns can also lead to artifacts. **(4)** *Improvement in shape correspondence:* Our use of Pinocchio leads to a limitation that the source and target shapes should be in approximately the same pose for skeleton embedding to succeed. The supplementary material shows some potential failure cases arising in *Transfer4D* when the assumptions are violated. In the future, we plan to incorporate pose-invariant shape correspondence modules building on [4] for additional supervision. **(5)** *LBS*: We currently use linear blend skinning(LBS) to deform the target mesh. However, LBS cannot capture non-rigid surface deformation. In future, we could replace the skinning module with alternatives such as SkinningNet [32] or neural blend shapes [22] . **(6)** *Data dependence:* As skinning decomposition relies on motion data, a limited motion range in the source could lead to an irregular skeleton, which may result in an unsuccessful transfer.

6. Conclusions

We presented *Transfer4D*, a first pipeline in the direction of unsupervised animation transfer from single-view commodity sensors to virtual characters modeled as polygonal mesh. In addition, we propose a novel skeletonization approach that helps in motion transfer reducing artifacts that arise solely through motion skeletonization. We also highlight the challenges faced and a few limitations of our and the existing approaches to be more effective for the democratization of animation transfer such as handling noisy data, pose, and large topological variations between source and target shapes.

7. Acknowledgements

We thank Srinidhi Hegde, Pranay Gupta, and Prof. Karan Singh for their insightful feedback and discussions.

References

[1] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)*, 39(4):62, 2020. 3

[2] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, jul 2003. 2, 3

[3] Raphael Anderegg, Loïc Ciccone, and Robert W. Sumner. Puppetphone: Puppeteering virtual characters using a smartphone. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, MIG '18, New York, NY, USA, 2018. Association for Computing Machinery. 3

[4] Oscar Kin-Chung Au, Chiew-Lan Tai, Daniel Cohen-Or, Youyi Zheng, and Hongbo Fu. Electors voting for fast automatic shape correspondence. In *Comput. Graph. Forum*, volume 29, pages 645–654. Citeseer, 2010. 2, 5, 8

[5] Andreas Bærentzen and Eva Rotenberg. Skeletonization via local separators. *ACM Trans. Graph.*, 40(5), sep 2021. 2, 5

[6] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72–es, July 2007. 2, 3, 4, 5, 7, 8

[7] Aljaz Bozic, Michael Zollhofer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7002–7012, 2020. 8

[8] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[9] N.D. Cornea, M.F. Demirci, D. Silver, Shokoufandeh, S.J. Dickinson, and P.B. Kantor. 3d object retrieval using many-to-many matching of curve skeletons. In *International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, pages 366–371, 2005. 2, 5

[10] Marvin Eisenberger, David Novotný, Gael Kerchenbaum, Patrick Labatut, Natalia Neverova, Daniel Cremers, and Andrea Vedaldi. Neuromorph: Unsupervised shape interpolation and correspondence in one go. *CoRR*, abs/2106.09431, 2021. 8

[11] Lin Gao, Jie Yang, Yi-Ling Qiao, Yukun Lai, Paul Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics*, 37(6):1–15, 2018. 2

[12] Pranay Gupta, Anirudh Thatipelli, Aditya Aggarwal, Shubh Maheshwari, Neel Trivedi, Sourav Das, and Ravi Kiran Sarvadevabhatla. Quo vadis, skeleton action recognition? *International Journal of Computer Vision*, May 2021. 2

[13] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005. 5

[14] Ming-Kai Hsieh, Bing-Yu Chen, and Ming Ouhyoung. Motion retargeting and transition in different articulated figures. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, pages 6 pp.–, 2005. 3

[15] Bae Jinseok, Jang Hojun, Min Cheol-Hui, Choi Hyungun, and Young Min Kim. Neural marionette: Unsupervised learning of motion skeleton and latent dynamics from volumetric video. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*, February 2022. 2, 7

[16] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, Sep 1978. 6

[17] Yanir Kleiman and Maks Ovsjanikov. Robust structure-based shape correspondence. In *Computer Graphics Forum*, volume 38, pages 7–20. Wiley Online Library, 2019. 2, 4

[18] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics (TOG)*, 33:1 – 10, 2014. 6, 7

[19] Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Trans. Graph.*, 28(5):1–10, dec 2009. 2

[20] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, page 1421–1430, Goslar, DEU, 2008. Eurographics Association. 4, 8

[21] Lei Li, Nicolas Donati, and Maks Ovsjanikov. Learning multi-resolution functional maps with spectral attention for robust shape matching. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 2

[22] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)*, 40(4):1, 2021. 8

[23] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 2, 3

[24] Zhouyingcheng Liao, Jimei Yang, Jun Saito, Gerard Pons-Moll, and Yang Zhou. Skeleton-free pose transfer for stylized 3d characters. In *European Conference on Computer Vision (ECCV)*. Springer, October 2022. 2

[25] Cheng Lin, Changjian Li, Yuan Liu, Nenglun Chen, Yi-King Choi, and Wenping Wang. Point2skeleton: Learning skeletal representations from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4277–4286, June 2021. 8

[26] Wenbin Lin, Chengwei Zheng, Jun-Hai Yong, and Feng Xu. Occlusionfusion: Occlusion-aware motion estimation for real-time dynamic 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1736–1745, June 2022. 2

[27] Or Litany, Tal Remez, Emanuele Rodolà, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5659–5667, 2017. 2

[28] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 2, 3

[29] Xuequan Lu, Zhigang Deng, Jun Luo, Wenzhi Chen, Sai-Kit Yeung, and Ying He. 3d articulated skeleton extraction using a single consumer-grade depth camera. *Computer Vision and Image Understanding*, 188:102792, 2019. 2, 7

[30] Martin Madaras, Michal Piovarči, Jana Běhal Dadová, Roman Franta, and Tomáš Kovačovský. Skeleton-based matching for animation transfer and joint detection. In *Proceedings of the 30th Spring Conference on Computer Graphics*, SCCG '14, page 91–98, New York, NY, USA, 2014. Association for Computing Machinery. 5

[31] Robin Magnet, Jing Ren, Olga Sorkine-Hornung, and Maks Ovsjanikov. Smooth non-rigid shape matching via effective dirichlet energy optimization. *arXiv preprint arXiv:2210.02870*, 2022. 2

[32] Albert Mosella-Montoro and Javier Ruiz-Hidalgo. Skinningnet: Two-stream graph convolutional neural network for skinning prediction of synthetic characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18593–18602, June 2022. 8

[33] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. 2

[34] Naoki Numaguchi, Atsushi Nakazawa, Takaaki Shiratori, and Jessica K. Hodgins. A puppet interface for retrieval of motion capture data. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, page 157–166, New York, NY, USA, 2011. Association for Computing Machinery. 3

[35] Pablo Rodríguez Palafox, Aljavz Bovzivc, Justus Thies, Matthias Nießner, and Angela Dai. Npms: Neural parametric models for 3d deformable shapes. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12675–12685, 2021. 2, 3

[36] Kàlmàn Palàgyi and Attila Kuba. A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, 19(7):613–627, 1998. 2

[37] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 2

[38] Helge Rhodin, James Tompkin, Kwang In Kim, Edilson de Aguiar, Hanspeter Pfister, Hans-Peter Seidel, and Christian Theobalt. Generalizing wave gestures from sparse examples for real-time character control. *ACM Trans. Graph.*, 34(6), oct 2015. 3

[39] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017. 2, 3

[40] Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. Unsupervised deep learning for structured shape matching. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1617–1627, 2019. 2

[41] Yeongho Seol, Carol O'Sullivan, and Jehee Lee. Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, page 213–221, New York, NY, USA, 2013. Association for Computing Machinery. 3

[42] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5474–5483, 2017. 2

[43] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[44] Olga Sorkine and Marc Alexa. As-Rigid-As-Possible Surface Modeling. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007. 4

[45] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, aug 2004. 2

[46] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80–es, jul 2007. 4

[47] Ramana Sundararaman, Gautam Pai, and Maks Ovsjanikov. Implicit field supervision for robust non-rigid shape matching. *arXiv preprint arXiv:2203.07694*, 2022. 2

[48] Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean curvature skeletons. *Comput. Graph. Forum*, 2012. 2

[49] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph.*, 28(3), jul 2009. 2, 5, 8

[50] Yu Tao, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *The IEEE International Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 7287–7296, 06 2018. 2

[51] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video, 2020. 2

[52] Dimitrios Tzionas and Juergen Gall. Reconstructing articulated rigged models from rgb-d videos. In *European Conference on Computer Vision Workshops 2016 (ECCVW'16) - Workshop on Recovering 6D Object Pose (R6D'16)*, 2016. 2, 8

[53] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011. 2

[54] Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409–409, 1965. 6

[55] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020. 2

[56] 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)*, 2019. 2, 7

[57] Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. Morig: Motion-aware rigging of character meshes from point clouds. In *Proc. ACM SIGGRAPH ASIA*, 2022. 2

[58] Katsu Yamane, Yuka Ariki, and Jessica Hodgins. Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, page 169–178, Goslar, DEU, 2010. Eurographics Association. 3

[59] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *CVPR*, 2021. 2

[60] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Ce Liu, and Deva Ramanan. Viser: Video-specific surface embeddings for articulated 3d shape reconstruction. In *NeurIPS*, 2021. 2, 3

[61] 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, 3

[62] Jie Yang, Lin Gao, Yu-Kun Lai, Paul L. Rosin, and Shihong Xia. Biharmonic deformation transfer with automatic key point selection. *Graphical Models*, 98:1–13, 2018. 2

[63] Li Yang, Takehara Hikari, Taketomi Takafumi, Zheng Bo, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. *IEEE International Conference on Computer Vision (ICCV)*, 2021. 4, 6, 7

[64] Quanshi Zhang, Xuan Song, Xiaowei Shao, Ryosuke Shibasaki, and Huijing Zhao. Unsupervised skeleton extraction and motion capture from 3d deformable matching. *Neurocomputing*, 100:170 – 182, 2013. Special issue: Behaviours in video. 2, 7

[65] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 3