NASA
Neural Articulated Shape Approximation

Boyang Deng[1], JP Lewis[1], Timothy Jeruzalski[1], Gerard Pons-Moll[2], Geoffrey Hinton[1], Mohammad Norouzi[1], and Andrea Tagliasacchi[1,3]

[1] Google Research
[2] MPI for Informatics, Saarland Informatics Campus, Germany
[3] University of Toronto, Canada

Abstract. Efficient representation of articulated objects such as human bodies is an important problem in computer vision and graphics. To efficiently simulate deformation, existing approaches represent 3D objects using polygonal meshes and deform them using skinning techniques. This paper introduces neural articulated shape approximation (NASA), an alternative framework that enables representation of articulated deformable objects using neural indicator functions that are conditioned on pose. Occupancy testing using NASA is straightforward, circumventing the complexity of meshes and the issue of water-tightness. We demonstrate the effectiveness of NASA for 3D tracking applications, and discuss other potential extensions.

Keywords: 3D deep learning, neural object representation, articulated objects, deformation, skinning, occupancy, neural implicit functions.

1 Introduction

There has been a surge of recent interest in representing 3D geometry using implicit functions parameterized by neural networks [37,20,39,10]. Such representations are flexible, continuous, and differentiable. Neural implicit functions are useful for "inverse graphics" pipelines for scene understanding [54], as back propagation through differentiable representations of 3D geometry is often required. That said, neural models of *articulated* objects have received little attention. Articulated objects are particularly important to represent animals and humans, which are central in many applications such as computer games and animated movies, as well as augmented and virtual reality.

Although parametric models of human body such as SMPL [33] have been integrated into neural network frameworks for self-supervision [27,38,40,53], these approaches depend heavily on polygonal mesh representations. Mesh representations require expert supervision to construct, and are not flexible for capturing topology variations. Furthermore, geometric representations often should fulfill several purposes simultaneously such as modeling the surface for rendering, or

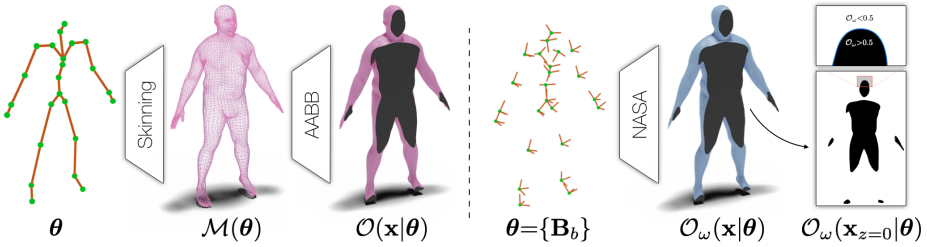

Fig. 1: **Teaser** – (left) Traditional articulated models map pose parameters $\boldsymbol{\theta}$ to a polygonal mesh $\mathcal{M}(\boldsymbol{\theta})$ via linear blend skinning; if one desires to *query* the occupancy of this representation, acceleration data structures need to be computed (e.g. axis align bounding box tree). (right) Conversely, NASA learns an *implicit* neural occupancy $\mathcal{O}_\omega$, which can be queried directly.

representing the volume to test intersections with the environment, which are not trivial when polygonal meshes are used [24]. Although neural models have been used in the context of articulated deformation [3], they *relegate* query execution to classical acceleration data structures, thus sacrificing full differentiability.

Our method represents articulated objects with a neural model, which outputs a differentiable occupancy of the articulated body in a specific pose. Like previous geometric learning efforts [37,12,39,9], we represent geometry by *indicator functions* – also referred to as occupancy functions – that evaluate to 1 inside the object and 0 otherwise. Unlike previous approaches, which focused on collections of static objects described by (unknown) shape parameters, we look at learning indicator functions as we vary *pose parameters*, which will be discovered by training on animation sequences. We show that existing methods [37,39,9] cannot encode pose variation reliably, because it is hard to learn the occupancy of every point in space as a function of a latent pose vector.

Instead, we introduce NASA, a neural decoder that exploits the structure of the underlying deformation driving the articulated object. Exploiting the fact that 3D geometry in *local* body part coordinates does not significantly change with pose, we classify the occupancy of 3D points as seen from the coordinate frame of each part. Our main architecture combines a collection of per-part learnable indicator functions with a per-part pose encoder to model localized non-rigid deformations. This leads to a significant boost in generalization to unseen poses, while retaining the useful properties of existing methods: differentiability, ease of spatial queries such as intersection testing, and continuous surface outputs. To demonstrate the flexibility of NASA, we use it to track point clouds by finding the maximum likelihood estimate of the pose under NASA's occupancy model. In contrast to mesh based trackers which are complex to implement, our tracker requires a few lines of code and is fully differentiable. Overall, our contributions include:

1. We propose a neural model of articulated objects to predict differentiable occupancy as a function of pose – the core idea is to model shapes by networks that encode a piecewise decomposition;
2. The results on learning 3D body deformation outperform previous geometric learning algorithms [39,9,39], and our surface reconstruction accuracy approaches that of mesh-based statistical body models [33];
3. The differentiable occupancy supports constant-time queries (.06 ms/query on an NVIDIA GTX 1080), avoiding the need to convert to separate representations, or the dynamic update of spatial acceleration data structures;
4. We derive a technique that employs occupancy functions for tracking 3D geometry via an implicit occupancy template, without the need to ever compute distance functions.

2 Related work

Neural articulated shape approximation provides a single framework that addresses problems that have previously been approached separately. The related literature thus includes a number of works across several different research topics.

Skinning algorithms. Efficient articulated deformation is traditionally accomplished with a skinning algorithm that deforms vertices of a mesh surface as the joints of an underlying abstract skeleton change. The classic linear blend skinning (LBS) algorithm expresses the deformed vertex as a weighted sum of that vertex rigidly transformed by several adjacent bones; see [23] for details. LBS is widely used in computer games, and is a core ingredient of some popular vision models [33]. Mesh sequences of general (not necessarily articulated) deforming objects have also been represented with skinning for the purposes of compression and manipulation, using a collection of non-hierarchical "bones" (i.e. transformations) discovered with clustering [25,30]. LBS has well-known disadvantages: the deformation has a simple algorithmic form that cannot produce pose-dependent detail, it results in characteristic volume-loss effects such as the "collapsing elbow" and "candy wrapper" artifacts [31, Figs. 2,3], and for best results the weights must be *manually* painted by artists. It is possible to add pose-dependent detail with a shallow or deep net regression [31,3], but this process operates as a correction to classical LBS deformation.

Object intersection queries. Registration, template matching, 3D tracking, collision detection, and other tasks require efficient inside/outside tests. A disadvantage of polygonal meshes is that they do not efficiently support these queries, as meshes often contain thousands of individual triangles that must be tested for each query. This has led to the development of a variety of spatial data structures to accelerate point-object queries [32,43], including voxel grids, octrees, kdtrees, and others. In the case of deforming objects, the spatial data structure must be repeatedly rebuilt as the object deforms. A further problem is that typically meshes may be constructed (or deformed) without regard to being "watertight" and thus do not have a clearly defined interior [24].

Part-based representations. For object intersection queries on articulated objects, it can be more efficient to approximate the overall shape in terms of a

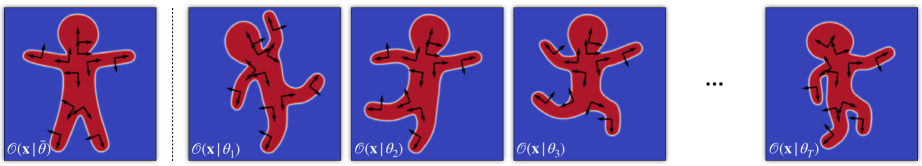

Fig. 2: **Notation** – (left) The ground truth occupancy $\mathcal{O}(\mathbf{x}|\bar{\boldsymbol{\theta}})$ in the rest frame and the pose parameters $\bar{\boldsymbol{\theta}} \equiv \{\bar{\mathbf{B}}_b\}_{b=1}^{B}$ representing the transformations of B bones. (right) T frames of an animation associated with pose parameters $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$ with corresponding occupancy $\{\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}_t)\}_{t=1}^{T}$; each $\boldsymbol{\theta}_t$ encodes the transformations of B bones. Note we shorthand $\{*_y\}$ to indicate an ordered set $\{*_y\}_{y=1}^{Y}$.

moving collection of rigid parts, such as spheres or ellipsoids, that support efficient querying [41]; see **supplementary material** for further discussion. Unfortunately this has the drawback of introducing a second approximate representation that does not exactly match the originally desired deformation. A further core challenge, and subject of continuing research, is the automatic creation of such *part-based* representations [1,19,21]. Unsupervised part discovery has been recently tackled by a number of deep learning approaches [12,34,8,13,17]. In general these methods address analysis and correspondence across shape collections, but do not target accurate representations of articulated objects, and do not account for pose-dependent deformations.

Neural implicit object representation. Finally, several recent works represent objects with neural implicit functions [37,9,39]. These works focus on the neural representation of *static* shapes in an *aligned* canonical frame and do not target the modeling of transformations. Our core contributions are to show that these architectures have difficulties in representing complex and detailed *articulated* objects (e.g. human bodies), and that a simple architectural change can address these shortcomings. Comparisons to these closely related works will be revisited in more depth in Section 6.

3 Neural Articulated Shape Approximation

This paper investigates the use of neural networks and implicit functions for modeling articulated shapes in $\mathbb{R}^d$. Let $\boldsymbol{\theta}$ denotes a vector representing the pose of an articulated shape, and let $\mathcal{O}:\mathbb{R}^d \rightarrow \{0,1\}$ denotes an occupancy function defining the exterior and interior of an articulated body. We are interested in modeling the joint distribution of pose and occupancy, which can be decomposed using the chain rule into a conditional occupancy term, and a pose prior term:

$$p(\boldsymbol{\theta}, \mathcal{O}) = p(\mathcal{O}|\boldsymbol{\theta})\, p(\boldsymbol{\theta}) \tag{1}$$

This paper focuses on building an expressive model of $p(\mathcal{O}|\boldsymbol{\theta})$, that is, occupancy conditioned on pose. Figure 2 illustrates this problem for d=2, and clarifies the notation. There is extensive existing research on pose priors $p(\boldsymbol{\theta})$ for

human bodies and other articulated objects [48,4,27]. Our work is orthogonal to such prior models, and any parametric or non-parametric $p(\boldsymbol{\theta})$ can be combined with our $p(\mathcal{O}|\boldsymbol{\theta})$ to obtain the joint distribution $p(\boldsymbol{\theta}, \mathcal{O})$. We delay the discussion of pose priors until Section 5.2, where we define a particularly simple prior that nevertheless supports sophisticated tracking of moving humans.

In what follows we describe different ways of building a pose conditioned occupancy function, denoted $\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta})$, which maps a 3D point $\mathbf{x}$ and a pose $\boldsymbol{\theta}$ onto a real valued occupancy value. Our goal is to learn a parametric occupancy $\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta})$ that mimics a ground truth occupancy $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$ as closely as possible, based on the following probabilistic interpretation:

$$p(\mathcal{O}|\boldsymbol{\theta}) \propto \prod_{\mathbf{x}\in\mathbb{R}^d} \exp\{-(\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) - \mathcal{O}(\mathbf{x}|\boldsymbol{\theta}))^2\} \ , \tag{2}$$

where we assume a standard normal distribution around the predicted real valued occupancy $\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta})$ to score an occupancy $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$.

We are provided with a collection of T ground-truth occupancies $\{\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}_t)\}_{t=1}^T$ associated with T poses. With a slight abuse of notation, we will henceforth use $\mathbf{x}$ to represent both a vector in $\mathbb{R}^d$, and its $\mathbb{R}^{d+1}$ homogeneous representation $[\mathbf{x}; 1]$. In our formulation, each pose parameter $\boldsymbol{\theta}$ represents a set of B *posed* bones/transformations, i.e., $\theta \equiv \{\mathbf{B}_b\}_{b=1}^B$. To help disambiguate the part-whole relationship, we also assume that for each mesh vertex $v \in \mathbf{V}$, the body part associations $\mathbf{w}(v)$ are available, where $\mathbf{w}(v) \in [0,1]^B$ with $\|\mathbf{w}(v)\|_1 = 1$.

Given pose parameters $\boldsymbol{\theta}$, we desire to query the corresponding indicator function $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$ at a point $\mathbf{x}$. This task is more complicated than might seem, as in the general setting this operation requires the computation of generalized winding numbers to resolve ambiguous configurations caused by self-intersections and non-necessarily watertight geometry [24]. However, when given a database of poses $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_t\}_{t=1}^T$ and corresponding *ground truth* indicator $\{\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}_t)\}_{t=1}^T$, we can formulate our problem as the minimization of the objective:

$$\mathcal{L}_{\text{occupancy}}(\omega) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \left[(\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) - \mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}))^2 \right] \tag{3}$$

where $p(\mathbf{x})$ is a density representing the sampling distribution of points in $\mathbb{R}^d$ (Section 4.4) and $\mathcal{O}_\omega$ is a neural network with parameters ω that represents our *neural articulated shape approximator*. We adopt a sampling distribution $p(\mathbf{x})$ that randomly samples in the volume surrounding a posed character, along with additional samples in the vicinity of the deformed surface.

4 Pose conditioned occupancy $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$

We investigate several neural architectures for the problem of articulated shape approximation; see Figure 3. We start by introducing an unstructured architecture (U) in Section 4.1. This baseline variant does not explicitly encode the knowledge of articulated deformation. However, typical articulated deformation

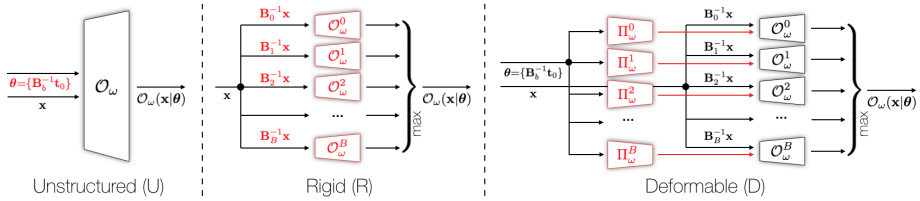

Fig. 3: The three architectures for $p(\mathcal{O}|\boldsymbol{\theta})$. The unstructured model employs a global MLP conditioned on pose, the rigid model expresses geometry as a composition of B *rigid* elements, while the deformable model via a composition of B *deformable* elements; we highlight the differences between models in red.

models [33] express deformed mesh vertices $\mathbf{V}$ reusing the information stored in rest vertices $\bar{\mathbf{V}}$. Hence, we can assume that computing the function $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$ in the deformed pose can be done by reasoning about the information stored at rest pose $\mathcal{O}(\mathbf{x}|\bar{\boldsymbol{\theta}})$. Taking inspiration from this observation, we investigate two different architecture variants, one that models geometry via a *piecewise-rigid* assumption (Section 4.2), and one that relaxes this assumption and employs a *quasi-rigid* decomposition, where the shape of each element can deform according to the pose (Section 4.3); see Figure 4.

4.1 Unstructured model – "U"

Recently, a series of papers [9,39,37] tackled the problem of modeling occupancy across shape datasets as $\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a latent code learned to encode the shape. These techniques employ deep and fully connected networks, which one can adapt to our setting by replacing the shape $\boldsymbol{\beta}$ with pose parameters $\boldsymbol{\theta}$, and using a neural network that takes as input $[\mathbf{x}, \boldsymbol{\theta}]$. Leaky ReLU activations are used for inner layers of the neural net and a sigmoid activation is used for the final output so that the occupancy prediction lies in the $[0, 1]$ range.

To provide pose information to the network, one can simply concatenate the set of affine bone transformations to the query point to obtain $[\mathbf{x}, \{\mathbf{B}_b\}]$ as the input. This results in an input tensor of size $3{+}16{\times}B$. Instead, we propose to represent pose as $\{\mathbf{B}_b^{-1}\mathbf{t}_0\}$, where $\mathbf{t}_0$ is the translation vector of the *root* bone in homogeneous coordinates, resulting in a smaller input of size $3{+}3{\times}B$; we ablate this choice against other alternatives in the **supplementary material**. Our unstructured baseline takes the form:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \mathrm{MLP}_\omega(\mathbf{x}, \underbrace{\{\mathbf{B}_b^{-1}\mathbf{t}_0\}}_{\text{pose}}) \tag{4}$$

Fig. 4: Our NASA representation models an articulated object as a collection of *deformable* components. The shape of each component is controlled by the pose of the subject, in a way that take inspiration from pose-space correctives [31].

4.2 Piecewise rigid model – "R"

The simplest structured deformation model for articulated objects assumes objects can be represented via a *piecewise rigid* composition of elements; e.g. [41,36]:

$$\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\mathcal{O}^b(\mathbf{x}|\boldsymbol{\theta})\} \tag{5}$$

We observe that if these elements are related to corresponding rest-pose elements through the rigid transformations $\{\mathbf{B}_b\}$, then it is possible to *query* the corresponding rest-pose indicator as:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\bar{\mathcal{O}}^b_\omega(\mathbf{B}_b^{-1}\mathbf{x})\} \tag{6}$$

where, similar to (4), we can represent each of components via a *learnable* indicator $\bar{\mathcal{O}}^b_\omega(.)$=MLP$^b_\omega(.)$. This formulation assumes that the local shape of each learned bone component stays *constant* across the range of poses when viewed from the corresponding coordinate frame, which is only a crude approximation of the deformation in realistic characters, and other deformable shapes.

4.3 Piecewise deformable model – "D"

We can generalize our models by combining the model of (4) to the one in (6), hence allowing the shape of each element to be *adjusted* according to pose:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\bar{\mathcal{O}}^b_\omega(\underbrace{\mathbf{B}_b^{-1}\mathbf{x}}_{\text{query}}|\boldsymbol{\theta})\} \tag{7}$$

Similar to (6) we use a *collection* of learnable indicator functions in rest pose $\{\mathcal{O}^b_\omega\}$, and to encode pose conditionals we take inspiration from (4). More specifically, we express our model as:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\bar{\mathcal{O}}^b_\omega(\mathbf{B}_b^{-1}\mathbf{x}, \underbrace{\mathit{\Pi}^b_\omega\left[\{\mathbf{B}_b^{-1}\mathbf{t}_0\}\right]}_{\text{part-specific pose}})\} \tag{8}$$

Similarly to (6), we model $\bar{\mathcal{O}}^b_\omega(.)$ via dense layers MLP$^b_\omega : \mathbb{R}^{3+D} \to \mathbb{R}$. The operator $\mathit{\Pi}^b_\omega : \mathbb{R}^{B\times 3} \to \mathbb{R}^D$ is a learnable *linear* subspace projection – one per each bone b. This choice is driven by the intuition that in typical skinned deformation models only *small subset* of the coordinate frames affect the deformation of a part. We employ D=4 throughout, see ablations in the supplementary material. Our experiments reveal that this bottleneck greatly improves generalization.

4.4 Technical details

The overall training loss for our model is:

$$\mathcal{L}(\omega) = \mathcal{L}_{\text{occupancy}}(\omega) + \lambda \mathcal{L}_{\text{weights}}(\omega) \tag{9}$$

where λ=$5e^{-1}$ was found through hyper-parameter tuning. We now detail the weights auxiliary loss, the architecture backbones, and the training procedure.

Auxiliary loss – skinning weights. As most deformable models are equipped with skinning weights, we exploit this additional source of information to facilitate learning of the part-based models (i.e. "R" and "D"). We label each mesh vertex $\mathbf{v}$ with the index of the corresponding highest skinning weight value $b^*(v)$= $\arg\max_b w(v)[b]$, and use the loss:

$$\mathcal{L}_{\text{weights}}(\omega) = \tfrac{1}{V}\tfrac{1}{B} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{\mathbf{v}} \sum_{b} \left(\bar{\mathcal{O}}^b_\omega(\mathbf{v}|\boldsymbol{\theta}) - \mathcal{I}_b(\mathbf{v})\right)^2 \tag{10}$$

where $\mathcal{I}_b(\mathbf{v})$=0.5 when b=b^*, and $\mathcal{I}_b(\mathbf{v})$=0 otherwise – recall that by convention the 0.5 level set is the surface represented by the occupancy function. Without such a loss, we could end up in the situation where a single (deformable) part could end up being used to describe the entire deformable model, and the trivial solution (zero) would be returned for all other parts.

Network architectures. To keep our experiments comparable across baselines, we use the same network architecture for all the models while varying the *width* of the layers. The network backbone is similar to DeepSDF [39], but simplified to 4 layers. Each layer has a residual connection, and uses the Leaky ReLU activation function with the leaky factor 0.1. All layers have the *same* number of neurons, which we set to 960 for the unstructured model and 40 for the structured ones. For the piecewise (6) and deformable (8) models the neurons are distributed across B=24 different channels (note $B\times 40 = 960$). Similar to the use of grouped filters/convolutions [29,22], such a structure allows for significant performance boosts compared to unstructured models (4), as the different branches can be executed in *parallel* on separate compute devices.

Training. All models are trained with the Adam optimizer, with batch size 12 and learning rate $1e-4$. For better gradient propagation, we use *softmax* whenever a max was employed in our expressions. For each optimization step, we use 1024 points sampled uniformly within the bounding box and 1024 points sampled near the ground truth surface. We also sample 2048 vertices out of 6890 mesh vertices at each step for $\mathcal{L}_{weights}$. The models are trained for 200K iterations for approximately 6 hours on a single NVIDIA Tesla V100.

5 Dense articulated tracking

Following the probabilistic interpretation of Section 3, we introduce an application of NASA to dense articulated 3D *tracking*; see [47]. Note that this section does not claim to beat the state-of-the-art in tracking of deformable objects [44,51,52], but rather it seeks to show *how* neural occupancy functions

can be used effectively in the development of dense tracking techniques. Taking the negative log of the joint probability in (1), the tracking problems can be expressed as the minimization of a pair of energies [47]:

$$\underset{\boldsymbol{\theta}^{(t)}}{\arg\min} \;\; E_{\text{fit}}(\mathbf{D}^{(t)}, \boldsymbol{\theta}^{(t)}) + E_{\text{prior}}(\boldsymbol{\theta}^{(t)}) \tag{11}$$

where $\mathbf{D}=\{\mathbf{x}_n\}_{n=1}^{N}$ is a point cloud in $\mathbb{R}^d$, and the superscript (t) indicates the point cloud and the pose associated with the t^{th} frame. The optimization for $\boldsymbol{\theta}^{(t)}$ is initialized with the minimizer computed at frame $(t-1)$. We also assume $\boldsymbol{\theta}^{(0)}$ is provided as ground truth, but discriminative models could also be employed to obtain an initialization [45,27]. In what follows, we often drop the (t) superscript for clarity of notation. We now discuss different aspects of this problem when an implicit representation of the model is used, including the implementation of fitting (Section 5.1) and prior (Section 5.2) energies, as well as details about the iterative optimization scheme (Section 5.3).

5.1 Fitting energy

If we could compute the Signed Distance Function (SDF) Φ of an occupancy $\mathcal{O}$ at a query point $\mathbf{x}$, then the fitness of $\mathcal{O}$ to input data could be measured as:

$$E_{\text{fit}}(\mathbf{D}, \boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathbf{D}} \|\Phi(\mathbf{x}|\mathcal{O}, \boldsymbol{\theta})\|^2 \tag{12}$$

The time complexity of computing SDF from an occupancy $\mathcal{O}$ that is discretized on a grid is *linear* in the number of voxels [16]. However, the number voxels grows as $O(n^d)$, making naive SDF computation impractical for high resolutions (large n) or high dimensions (in practice, $d\geq 3$ is already problematic). Spatial acceleration data structures (kdtrees and octrees) are commonly employed, but these data structures still require an overall $O(n\log(n))$ pre-processing (where n is the number of polygons), and they need to be re-built at every frame (as $\boldsymbol{\theta}$ changes), and do not support implicit representations.

Recently, Dou et al. [14] proposed to smooth an occupancy function with a Gaussian blur kernel to *approximate* Φ in the near field of the surface. Following this idea, our fitting energy can be re-expressed as:

$$E_{\text{fit}}(\mathbf{D}, \boldsymbol{\theta}) = \sum_{\mathbf{x}\in\mathbf{D}} \|\mathcal{N}_{0,\sigma^2} \circledast \mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) - 0.5\|^2 \tag{13}$$

where $\mathcal{N}_{0,\sigma^2}$ is a Gaussian kernel with a zero mean and a variance σ^2, and $\circledast$ is the convolution operator. This approximation is suitable for tracking, as large values of distance should be associated with *outliers* in a registration optimization [6], and therefore ignored. Further, this approximation can be explained via the algebraic relationship between heat kernels and distance functions [11]. Note that we *intentionally* use $\mathcal{O}$ instead of $\mathcal{O}_\omega$, as what follows is applicable to *any* implicit representation, not just our neural occupancy $\mathcal{O}_\omega$.

Dou et al. [14] used (13) being given a voxelized representation of $\mathcal{O}$, and relying on GPU implementations to efficiently compute 3D convolutions $\circledast$. To circumvent these issues, we re-express the convolution via stochastic sampling:

$$\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) \circledast \mathcal{N}_{0,\sigma^2} = \int \mathcal{O}(\mathbf{s}|\boldsymbol{\theta}) g(\mathbf{x}-\mathbf{s}|0,\sigma^2)\, d\mathbf{s} \quad \text{(definition of convolution)} \tag{14}$$

$$= \int \mathcal{O}(\mathbf{s}|\boldsymbol{\theta}) g(\mathbf{s}-\mathbf{x}|0,\sigma^2)\, d\mathbf{s} \quad \text{(symmetry of Gaussian)} \tag{15}$$

$$= \int \mathcal{O}(\mathbf{s}|\boldsymbol{\theta}) g(\mathbf{s}|\mathbf{x},\sigma^2)\, d\mathbf{s} \quad \text{(definition of Gaussian)} \tag{16}$$

$$= \mathbb{E}_{\mathbf{s}\sim\mathcal{N}_{\mathbf{x},\sigma^2}}\left[\mathcal{O}(\mathbf{s}|\boldsymbol{\theta})\right] \quad \text{(definition of expectation)} \tag{17}$$

Overall, Equation 17 allows us to design a tracking solution that directly operates on occupancy functions, *without* the need to compute signed distance functions [48], closest points [47], or 3D convolutions [14]. It further provides a direct cost/accuracy control in terms of the number of samples used to approximate the expectation in (17). However, the gradients $\nabla_{\boldsymbol{\theta}}$ of (17) also need to be available – we achieve this by applying the *re-parameterization* trick [28] to (17):

$$\nabla_{\boldsymbol{\theta}}\left[\mathbb{E}_{\mathbf{s}\sim\mathcal{N}_{\mathbf{x},\sigma^2}}\left[\mathcal{O}(\mathbf{s}|\boldsymbol{\theta})\right]\right] = \mathbb{E}_{\mathbf{s}\sim\mathcal{N}_{0,1}}\left[\nabla_{\boldsymbol{\theta}}\mathcal{O}(\mathbf{x}+\sigma\mathbf{s}|\boldsymbol{\theta})\right] \tag{18}$$

5.2 Pose prior energy

An issue of generative tracking is that once the model is too far from the target (e.g. fast motion) there will be no proper gradient to correct it. If we directly optimize for transformation without any constraints, there is a high chance that the model will degenerate into such a case. To address this, we impose a prior:

$$E_{\text{prior}}(\boldsymbol{\theta}{=}\{\mathbf{B}_b\}) = \sum_{(b_1,b_2)\in\mathcal{E}} \left\|(\bar{\mathbf{t}}_{b_2} - \bar{\mathbf{t}}_{b_1}) - \mathbf{B}_{b_1}^{-1}\mathbf{t}_{b_2}\right\|_2^2 \tag{19}$$

where $\mathcal{E}$ is the set of directed edges (b_1, b_2) on the pre-defined directed rig with b_1 as the parent, and recall $\mathbf{t}_b$ is the translation vector of matrix $\mathbf{B}_b$. One can view this loss as aligning the vector pointing to t_{b_2} at run-time with the vector at rest pose, i.e. $(\bar{\mathbf{t}}_{b_2} - \bar{\mathbf{t}}_{b_1})$. We emphasize that more sophisticated priors exist, and could be applied, including employing a hierarchical skeleton [48], or modeling the density of joint angles [4]. The simple prior used here is chosen to highlight the effectiveness of our neural occupancy model *independent* of such priors.

5.3 Iterative optimization

One would be tempted to use the gradients of (13) to track a point cloud via *iterative* optimization. However, it is known that when optimizing rotations *centering* the optimization about the current state is heavily advisable [47]. Indexing time

by (t) and given the update rule $\boldsymbol{\theta}^{(t)}{=}\boldsymbol{\theta}^{(t-1)}{+}\,\Delta\boldsymbol{\theta}^{(t)}$, the iterative optimization of (13) can be expressed as:

$$\underset{\Delta\boldsymbol{\theta}^{(t)}}{\arg\min} \sum_{\mathbf{x}\in\mathbf{D}^{(t)}} \left\| \mathbb{E}_{\mathbf{s}\sim\mathcal{N}_{\mathbf{x},\sigma^2}} \left[\mathcal{O}_\omega(\mathbf{s}|\boldsymbol{\theta}^{(t-1)} + \Delta\boldsymbol{\theta}^{(t)}) \right] - 0.5 \right\|^2 \tag{20}$$

where in what follows we omit the index (t) for brevity of notation. As the pose $\boldsymbol{\theta}$ is represented by matrices, we represent the transformation differential as:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) = \mathcal{O}_\omega(\mathbf{x}|\{(\mathbf{B}_b\Delta\mathbf{B}_b)^{-1}\}) = \mathcal{O}_\omega(\mathbf{x}|\{\Delta\mathbf{B}_b^{-1}\mathbf{B}_b^{-1}\}), \tag{21}$$

resulting in the optimization:

$$\underset{\{\Delta\mathbf{B}_b^{-1}\}}{\arg\min} \sum_{\mathbf{x}\in\mathbf{D}} \left\| \mathbb{E}_{\mathbf{s}\sim\mathcal{N}_{\mathbf{x},\sigma^2}} \left[\mathcal{O}_\omega(\mathbf{s}|\{\Delta\mathbf{B}_b^{-1}\mathbf{B}_b^{-1}\}) \right] - 0.5 \right\|^2 \tag{22}$$

where we parameterize the rotational portion of elements in the collection $\{\Delta\mathbf{B}_b^{-1}\}$ by two (initially orthogonal) vectors [57], and re-orthogonalize them *before* inversion within each optimization update $\mathbf{B}_b^{(i+1)}{=}\mathbf{B}_b^{(i)}(\Delta\mathbf{C}_b^{(i)})^{-1}$, where $\Delta\mathbf{C}_b{=}\Delta\mathbf{B}_b^{-1}$ are the quantities the solver optimizes for. In other words, we optimize for the *inverse* of the coordinate frames in order to avoid back-propagation through matrix inversion.

6 Results and discussion

We describe the training data (Section 6.1), quantitatively evaluate the performance of our neural 3D representation on several datasets (Section 6.2), as well as demonstrate its usability for tracking applications (Section 6.3). We conclude by contrasting our technique to recent methods for implicit-learning of geometry (Section 6.4). Ablation studies validating *each* of our technical choices can be found in the **supplementary material**.

6.1 Training data

Our training data consists of sampled indicator function values, transformation frames ("bones") per pose, and skinning weights. The samples used for training (3) come from two sources (each comprising a total of $100,000$ samples): ① we randomly sample points uniformly within a bounding box scaled to 110% of its original diagonal dimension; ② we perform Poisson disk sampling on the surface, and randomly displace these points with isotropic normal noise with $\sigma{=}.03$. The ground truth indicator function at these samples are computed by casting randomized rays and checking the *parity* (i.e. counting the number of intersections) – generalized winding numbers [24] or sign-agnostic losses [2] could also be used for this purpose. The test reconstruction performance is evaluated by comparing the predicted indicator values against the ground truth samples on the full set of $100,000$ samples. We evaluate using mean Intersection over Union (IoU), Chamfer-L1 [15] and F-score (F%) [50] with a threshold set to 0.0001. The meshes are obtained from the "DFaust" [5] and "Transitions" sub-datasets of AMASS [35], as detailed in Section 6.2.

Model	mIoU↑	Chamfer L1↓	F%↑
U	.702	.00631	46.15
R	.932	.00032	93.94
D	**.959**	**.00004**	**98.54**

Table 1: AMASS / DFaust

Model	mIoU↑	Chamfer L1↓	F%↑
U	.520	.01057	26.83
R	.936	.00006	96.71
D	**.965**	**.00002**	**99.42**

Table 2: AMASS / Transitions

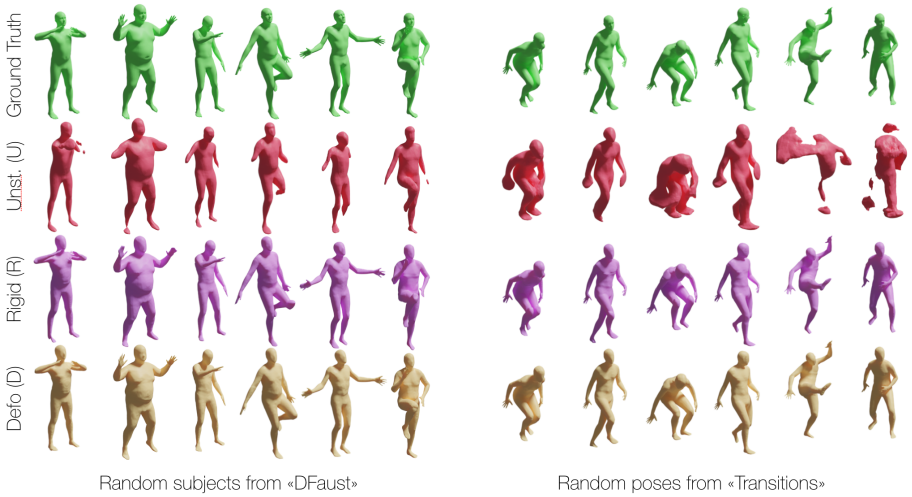

Fig. 5: The qualitative performance of our three models in reconstructing the occupancy function on the (left) DFaust and (right) Transitions dataset.

6.2 Reconstruction

We employ the "DFaust" portion of the AMASS dataset to verify that our model can be used effectively *across* different subjects. This dataset contains 10 subjects, 10 sequences/subject, and ≈300 frames/sequence on average. We train 100 different models by optimizing (9): for each subject we use 9 sequences for training, leaving one out for testing to compute our metrics. We average these metrics across the 100 runs, and report these in Table 1. Note how learning a deformable model via decomposition provides *striking* advantages, as quantified by the fact that the rigid (R) baseline is consistently better than the unstructured (U) baseline under *any* metric – a +**49**% in F-score. Similar improvements can be noticed by comparing the rigid (R) to the deformable (D) model, where the latter achieves an additional +**5**% in F-score. Figure 5 (second row) gives a qualitative visualization of how the unstructured models struggles in generalizing to poses that are sufficiently different from the ones in the training set.

We employ the "Transitions" portion of the AMASS dataset to further study the performance of the model when more training data and a larger diversity of motions) is available for a *single* subject. This dataset contains 110 sequences of one individual, with ≈1000+ frames/sequence. We randomly sample ≈250 frames from each sequence, randomly select 80 sequences for training, and keep the remaining 30 sequences for testing; see our **supplementary material**. Re-

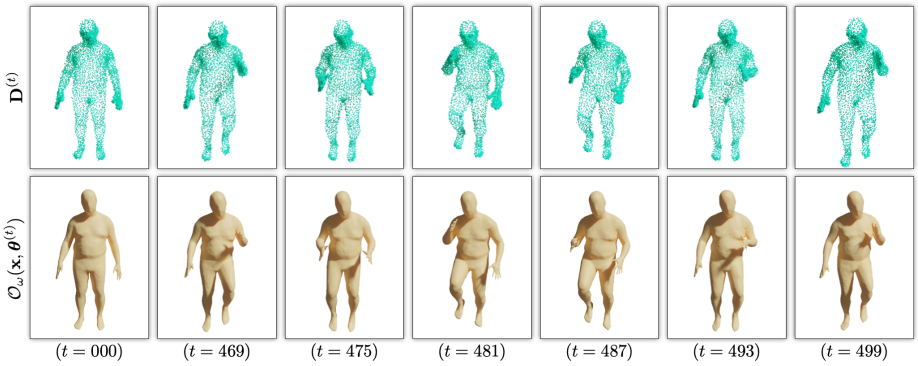

Fig. 6: A few frames of our neural model *tracking* the point cloud of the DFaust "hard" (02-09) sequence; these results can be better appreciated in our **video**.

$p(\mathcal{O}\|\boldsymbol{\theta})$	$p(\boldsymbol{\theta})$	⊛	mIoU↑	Chamfer L1↓	F%↑
U	✓	✓	.845	.00383	61.63
D	✓	✓	**.968**	**.00004**	**99.08**
oracle	–	–	.976	.00004	99.01

Table 3: DFaust "easy" (00-01)

$p(\mathcal{O}\|\boldsymbol{\theta})$	$p(\boldsymbol{\theta})$	⊛	mIoU↑	Chamfer L1↓	F%↑
U	✓	✓	.686	.00700	50.63
D	✓	✓	**.948**	**.00006**	**96.48**
oracle	–	–	.959	.00006	96.80

Table 4: DFaust "hard" (02-09)

sults are shown in Table 2. The conclusions are analogous to the ones we made from DFaust. Further, note that in this more difficult dataset containing a larger variety of more complex motions, the unstructured model struggles even more significantly ($U \rightarrow R$: $+\mathbf{70}\%$ in F-score). As the model is exposed to more poses compared to DFaust, the reconstruction performance is also improved. Moving from DFaust to Transitions results in a $+\mathbf{1}\%$ in F-score for the deformable model.

6.3 Tracking

We validate our tracking technique on two sequences from the DFaust dataset; see Figure 6. Note these are test sequences, and were *not* used to train our model. The prior $p(\boldsymbol{\theta})$ (Section 5.2) and the stochastic optimization ⊛ (Section 5.1) can be applied to both unstructured (U) and structured (D) representations, with the latter leading to significantly better tracking performance. The quantitative results reported for the "easy" (Table 3) and "hard" (Table 4) tracking sequences are best understood by watching our **supplementary video**. It is essential to re-state that we are not trying to beat traditional baselines, but rather seek to *illustrate* how NASA, once trained, can be readily used as a 3D representation for classical vision tasks. For the purpose of this illustration, we use only noisy panoptic point clouds (i.e. complete [14] rather than incomplete [48] data), and do not use any discriminative per-frame re-initializer as would typically be employed in a contemporary tracking system.

6.4 Discussion

The recent success of neural implicit representations of geometry, introduced by [9,39,37], has heavily relied on the fact that the geometry in ShapeNet datasets [7] is *canonicalized*: scaled to unit ranges and consistently oriented. Research has highlighted the importance of expressing information in a canonical frame [55], and one could interpret our method as a way to achieve this within the realm of articulated motion. To understand the shortcomings of unstructured models, one should remember that as an object moves, much of the local geometric details remain *invariant* to articulation. However, unstructured pose conditioned models are forced to *memorize* these details in any pose they seek to reconstruct. Hence, as one evaluates unstructured models *outside* of their training manifold, their performance *collapses* – as quantified by the +49% performance change as we move from unstructured to rigid models; see Table 1. One could also argue that given sufficient capacity, a neural network *should* be able to learn the *concept* of coordinate frames and transformations. However, multiplicative relationships between inputs (e.g. dot products) are difficult to learn for neural networks [26, Sec. 2.3]. As changes of coordinate frames are nothing but collections of dot products, one could use this reasoning to justify the limited performance of unstructured models. We conclude by clearly contrasting our method, targeting the modeling of $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$ to those that address shape completion $\mathcal{O}(\mathbf{x}|\mathbf{D})$ [18,56,42,2]. In contrast to these, our solution, to the best of our knowledge, represents the first attempt to create a "neural implicit rig" – from a computer graphics perspective – for articulated deformation modeling.

One limitation of our work is the reliance on $\{\bar{\mathbf{B}}_b\}$, which could be difficult to obtain in in-the-wild settings, as well as skinning weights to guide the part-decomposition; how to automatically regress these quantities from raw observations is an open problem. Our model is also currently limited to *individual* subjects, and to be competitive to mesh-based models one would also have to learn identity parameters (i.e. the $\boldsymbol{\beta}$ parameters of SMPL [33]). Finally, our representation currently fails to capture high frequency features (e.g. see the geometric details of the face region in Figure 5); however, recent research on implicit representations [49,46] can likely mitigate this issue.

7 Conclusions

We introduce a novel neural representation of a particularly important class of 3D objects: *articulated* bodies. We use a structured neural occupancy approach, enabling both direct occupancy queries and deformable surface representations that are competitive with classic hand-crafted mesh representations. The representation is fully differentiable, and enables tracking of realistic articulated bodies – traditionally a complex task – to be almost *trivially* implemented. Crucially, our work demonstrates the value of incorporating a task-appropriate inductive bias into the neural architecture. By acknowledging and encoding the quasi-rigid part structure of articulated bodies, we represent this class of objects with higher quality, and significantly better generalization.

References

1. Anguelov, D., Koller, D., Pang, H.C., Srinivasan, P., Thrun, S.: Recovering articulated object models from 3d range data. In: Uncertainty in Artificial Intelligence (2004) 4
2. Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. arXiv preprint arXiv:1911.10414 (2019) 11, 14
3. Bailey, S.W., Otte, D., Dilorenzo, P., O'Brien, J.F.: Fast and deep deformation approximations. SIGGRAPH (2018) 2, 3
4. Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., Black, M.J.: Keep it SMPL: Automatic estimation of 3d human pose and shape from a single image. In: ECCV (2016) 5, 10
5. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic FAUST: Registering human bodies in motion. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2017) 11
6. Bouaziz, S., Tagliasacchi, A., Pauly, M.: Sparse iterative closest point. In: SGP (2013) 9
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv:1512.03012 (2015) 14
8. Chen, Z., Yin, K., Fisher, M., Chaudhuri, S., Zhang, H.: Bae-net: Branched autoencoder for shape co-segmentation. In: ICCV (2019) 4
9. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. CVPR (2019) 2, 3, 4, 6, 14
10. Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3d shape reconstruction and completion. In: CVPR (2020) 1
11. Crane, K., Weischedel, C., Wardetzky, M.: Geodesics in heat: A new approach to computing distance based on heat flow. ACM TOG (2013) 9
12. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: Cvxnet: Learnable convex decomposition. CVPR (2020) 2, 4
13. Deng, B., Kornblith, S., Hinton, G.: Cerberus: A multi-headed derenderer. arXiv:1905.11940 (2019) 4
14. Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S.R., Kowdle, A., Escolano, S.O., Rhemann, C., Kim, D., Taylor, J., et al.: Fusion4d: Real-time performance capture of challenging scenes. ACM TOG (2016) 9, 10, 13
15. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017) 11
16. Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Theory of computing (2012) 9
17. Gao, L., Yang, J., Wu, T., Yuan, Y.J., Fu, H., Lai, Y.K., Zhang, H.: SDM-NET: deep generative network for structured deformable mesh. ACM TOG (2019) 4
18. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Deep structured implicit functions. CVPR (2019) 14
19. de Goes, F., Goldenstein, S., Velho, L.: A hierarchical segmentation of articulated bodies. In: SGP (2008) 4
20. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: Atlasnet: A papier-mâché approach to learning 3d surface generation. arXiv preprint arXiv:1802.05384 (2018) 1
21. Huang, Q., Koltun, V., Guibas, L.: Joint shape segmentation with linear programming. ACM TOG (2011) 4

22. Ioannou, Y., Robertson, D., Cipolla, R., Criminisi, A.: Deep Roots: Improving CNN efficiency with hierarchical filter groups. In: CVPR (2017) 8
23. Jacobson, A., Deng, Z., Kavan, L., Lewis, J.: Skinning: Real-time shape deformation. In: ACM SIGGRAPH Courses (2014) 3
24. Jacobson, A., Kavan, L., Sorkine-Hornung, O.: Robust inside-outside segmentation using generalized winding numbers. ACM TOG (2013) 2, 3, 5, 11
25. James, D.L., Twigg, C.D.: Skinning mesh animations. SIGGRAPH (2005) 3
26. Joseph-Rivlin, M., Zvirin, A., Kimmel, R.: Momen(e)t: Flavor the moments in learning to classify shapes. In: CVPR Workshops (2019) 14
27. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: CVPR (2018) 1, 5, 9
28. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013) 10
29. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012) 8
30. Le, B.H., Deng, Z.: Smooth skinning decomposition with rigid bones. ACM TOG (2012) 3
31. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In: SIGGRAPH (2000) 3, 7
32. Lin, M.C., Manocha, U.D., Cohen, J.: Collision detection: Algorithms and applications (1996) 3
33. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. SIGGRAPH Asia (2015) 1, 3, 6, 14
34. Lorenz, D., Bereska, L., Milbich, T., Ommer, B.: Unsupervised part-based disentangling of object shape and appearance. arXiv:1903.06946 (2019) 4
35. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. ICCV (2019) 11
36. Melax, S., Keselman, L., Orsten, S.: Dynamics based 3d skeletal hand tracking. In: Graphics Interface (2013) 7
37. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. arXiv:1812.03828 (2018) 1, 2, 4, 6, 14
38. Omran, M., Lassner, C., Pons-Moll, G., Gehler, P., Schiele, B.: Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In: International Conference on 3D Vision (3DV) (sep 2018) 1
39. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. CVPR (2019) 1, 2, 3, 4, 6, 8, 14
40. Pavlakos, G., Zhu, L., Zhou, X., Daniilidis, K.: Learning to estimate 3D human pose and shape from a single color image. In: CVPR (2018) 1
41. Remelli, E., Tkach, A., Tagliasacchi, A., Pauly, M.: Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In: ICCV (2017) 4, 7
42. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: CVPR (2019) 14
43. Samet, H.: Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley Longman Publishing Co., Inc. (1990) 3
44. Shen, J., Cashman, T.J., Ye, Q., Hutton, T., Sharp, T., Bogo, F., Fitzgibbon, A.W., Shotton, J.: The phong surface: Efficient 3d model fitting using lifted optimization (2020) 8

45. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR (2011) 9
46. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions (2020) 14
47. Tagliasacchi, A., Bouaziz, S.: Dynamic 2d/3d registration. Proc. Symposium on Geometry Processing (Technical Course Notes) (2018) 8, 9, 10
48. Tagliasacchi, A., Schröder, M., Tkach, A., Bouaziz, S., Botsch, M., Pauly, M.: Robust articulated-icp for real-time hand tracking. In: SGP (2015) 5, 10, 13
49. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739 (2020) 14
50. Tatarchenko, M., Richter, S.R., Ranftl, R., Li, Z., Koltun, V., Brox, T.: What do single-view 3d reconstruction networks learn? In: CVPR (2019) 11
51. Taylor, J., Tankovich, V., Tang, D., Keskin, C., Kim, D., Davidson, P., Kowdle, A., Izadi, S.: Articulated distance fields for ultra-fast tracking of hands interacting. ACM Transactions on Graphics (TOG) (2017) 8
52. Tkach, A., Tagliasacchi, A., Remelli, E., Pauly, M., Fitzgibbon, A.: Online generative model personalization for hand tracking. ACM Transaction on Graphics (Proc. SIGGRAPH Asia) (2017) 8
53. Tung, H.Y., Tung, H.W., Yumer, E., Fragkiadaki, K.: Self-supervised learning of motion capture. In: Advances in Neural Information Processing Systems. pp. 5236–5246 (2017) 1
54. Valentin, J., Keskin, C., Pidlypenskyi, P., Makadia, A., Sud, A., Bouaziz, S.: Tensorflow graphics: Computer graphics meets deep learning (2019) 1
55. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: CVPR (2019) 14
56. Xu, Q., Wang, W., Ceylan, D., Mech, R., Neumann, U.: DISN: Deep implicit surface network for high-quality single-view 3d reconstruction. In: NeurIPS (2019) 14
57. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR (2019) 11