NASA: Neural Articulated Shape Approximation

Timothy Jeruzalski* Boyang Deng* Mohammad Norouzi
JP Lewis Geoffrey Hinton Andrea Tagliasacchi

Google Research

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 objects as meshes and deform them using skinning techniques. This paper introduces neural articulated shape approximation (NASA), a framework that enables efficient representation of articulated deformable objects using neural indicator functions parameterized by pose. In contrast to classic approaches, NASA avoids the need to convert between different representations. For occupancy testing, NASA circumvents the complexity of meshes and mitigates the issue of water-tightness. In comparison with regular grids and octrees, our approach provides high resolution without high memory use.

1. Introduction

There has been a surge of recent interest in computer vision in developing better and more flexible 3D representations of objects and scenes [6, 11, 28, 29]. These recent advances are partly motivated by the development of "inverse graphics" pipelines for scene understanding [34]. With the dominance of deep neural networks in computer vision, we have seen inverse graphics flourish, especially when *differentiable* models of geometry are available. However, among possible applications, neural models of *articulated* objects have received little attention. Models of articulated objects are particularly important because they encompass 3D representations of humans. Virtual humans are a central subject not only in computer games and animated movies, but also in other applications such as augmented and virtual reality.

Existing geometric learning algorithms include self-supervised methods for face [32], body [18], and low level geometry [8], all relying on optimization of fully differentiable encoder-decoder architectures. The use of neural decoders is also a possibility [14], but the quality of results can receive a significant boost when more structure about the phenomena being modeled is directly expressed within the architecture; see [33] for an example. Geometric models often must fullfill several purposes such as representing the shape for rendering or representing the volume for the purpose of intersection queries. Although neural models have been used in the context of articulated deformation [2], they have addressed only deformations while *relegating* both intersection queries and the overall articulation to classic methods, thus sacrificing full differentiability.

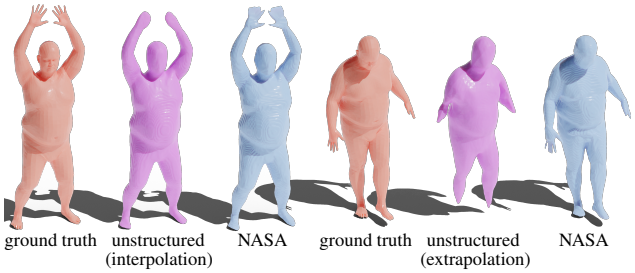

Figure 1: The occupancy of a ground truth mesh (red) can be represented by a network in an unstructured way (purple [4, 28, 29]) or by our NASA approximation (blue) that models the underlying quasi-rigid structure. Notice the stark difference in the performance when interpolating vs. extrapolating.

Our method represents articulated objects with a differentiable neural model. We train a neural decoder $\mathcal{D}_\omega$ that exploits the structure of the underlying deformation driving the articulated object. As with some previous geometric learning efforts [4, 8, 28, 29] we represent geometry by *indicator functions* – also referred to as occupancy functions – that evaluate to 1 inside the object and 0 otherwise. If desired, an explicit surface can be extracted via marching cubes [24]. 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. Overall, our contributions are:

1. We propose a way to approximate articulated deformable models via neural networks – the core idea is to model

*equal contributors

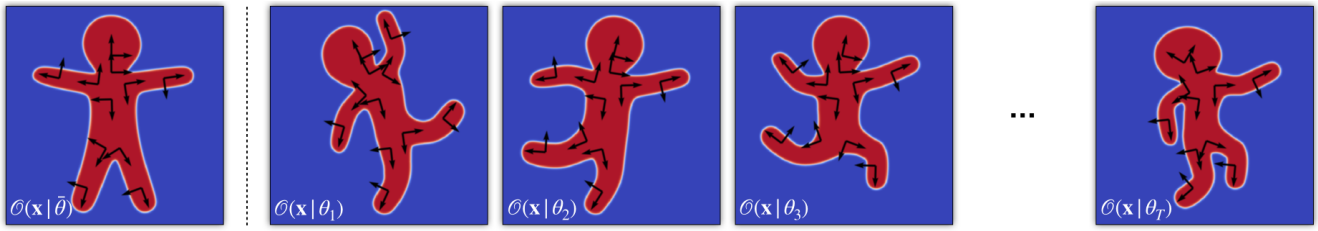

Figure 2: **Data and notation** – (left) The ground truth occupancy $\mathcal{O}(\mathbf{x}|\bar{\boldsymbol{\theta}})$ in the rest frame and the pose parameters $\bar{\boldsymbol{\theta}} \equiv \{\bar{\mathbf{B}}_b\}_{b=1}^{B}$ representing the transformations of B bones. (right) T frames of an animation 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.

shapes by networks that encode a [quasi] piecewise rigid decomposition;

2. We show how explicitly expressing structure of deformation in the network allows for fewer model parameters while providing both similar performance and better generalization;
3. The indicator function representation supports efficient intersection and collision queries, avoiding the need to convert to a separate representation for this purpose;
4. The results on learning 3D body motion outperform previous geometric learning algorithms [4, 29, 29] and are competitive with a hand-crafted statistical body model [23].

2. Related works

Neural 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 fields.

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 [16] for details. LBS is widely used in computer games, and is a core ingredient of popular vision models [23]. 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" (transformations) discovered with clustering [17, 20]. 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 [21, 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 deep net regression [2], 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 [22, 31], including voxel grids, octrees, 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 typical meshes may be constructed without regard to being "watertight" and thus do not have a *clearly* defined interior [15].

Part-based representations. For object intersection queries on articulated objects, it can be more efficient to approximate the overall shape in terms of a moving collection of rigid parts, such as spheres or ellipsoids, that support an efficient intersection test [30]. Unfortunately this has the drawbacks 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 this *part-based* representation [1, 7, 12]. Unsupervised part discovery has been recently tacked by a number of deep learning approaches [5, 8–10, 25]. In general these methods address analysis and correspondence across shape collections, and do not target accurate representations of articulated deforming objects. Pose-dependent deformation effects are also not considered in any of these approaches.

Neural implicit object representation. Finally, several recent works represent objects with neural *implicit* functions [4, 28, 29]. These works focus on the neural representation of *static* shapes in an *aligned* canonical frame and do not target the modeling of transformations. Our work can be considered an extension of these methods, where the core difference is its ability to efficiently represent complex and detailed *articulated* objects (e.g. human bodies).

Figure 3: **Qualitative / 2D –** Approximation quality across model architectures on the gingerbread dataset. We report unstructured (U), piecewise-rigid (R), and piecewise-deformable (U) models and indicate network width via @; see Section 4.4.

3. Neural Articulated Shapes Approximation

Figure 2 illustrates the problem of articulated shape approximation in 2D. We are provided with an articulated object in the *rest* pose (the typical *T-pose*) and the corresponding *occupancy* function $\mathcal{O}(\mathbf{x}|\bar{\boldsymbol{\theta}})$. In addition, 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. In our formulation, each pose parameter $\boldsymbol{\theta}$ represents a set of of *posed* transformations associated with B bones, 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 skinning weights $\mathbf{w}(v)$ are available, where $w(v) \in [0,1]^B$ with $\|\mathbf{w}(v)\|_1 = 1$.

Given the collection of 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 [15]. 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 following objective:

$$\mathcal{L}_{\text{occupancy}}(\omega) = \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\left(\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) - \mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) \right)^2 \right] \quad (1)$$

where $p(\mathbf{x})$ is a density representing the sampling distribution of points in $\mathbb{R}^3$ (Section 4.4) and $\mathcal{O}_\omega$ is a neural network with parameters ω that represents our *neural 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.

One can view $\mathcal{O}(\mathbf{x}|\boldsymbol{\theta})$ as a binary classifier that aims to separate the interior of the shape from its exterior. Accordingly, one can use a binary cross-entropy loss for optimization, but our preliminary experiments suggest that both L2 and cross-entropy losses perform similarly for shape approximation. Thus, we adopt (1) in our experiments.

4. Neural Architectures for NASA

We investigate several neural architectures for the problem of articulated shape approximation. The unstructured architecture in Section 4.1 does not explicitly encode the knowledge of articulated deformation. However, typical articulated deformation models [23] 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 $\bar{\mathcal{O}}(\mathbf{x}|\bar{\boldsymbol{\theta}})$. 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).

4.1. Unstructured model – "U"

Recently, a series of papers [4, 28, 29] 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 $[\mathbf{x}, \boldsymbol{\theta}]$. 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 is bounded between 0 and 1.

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 the composition of a query point $\mathbf{x}$ with a pose $\boldsymbol{\theta}$ via $\{\mathbf{B}_b^{-1}\mathbf{x}\}_{b=1}^{B}$, resulting in a smaller input of size $3\times B$. Our unstructured baseline takes the form:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \text{MLP}_\omega(\{\mathbf{B}_b^{-1}\mathbf{x}\}) \quad (2)$$

We term this the *unstructured* model as it *does not* explicitly model the underlying deformation process.

4.2. Piecewise rigid model – "R"

The simplest structured deformation model for articulated objects assumes our object can be represented via a *piece-*

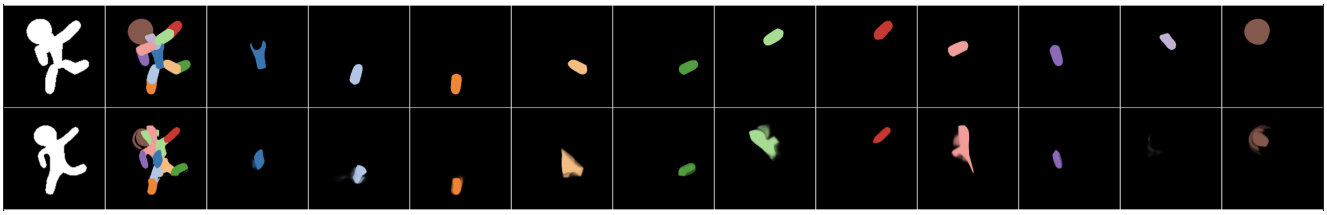

Figure 4: **Piecewise rigid model (R) –** is applied to (top) a piecewise rigid motion and (bottom) a linear blend skinned motion dataset. The first column shows the ground truth indicator values for the pose, whereas the second column is the predicted indicator. The indicator is constructed by taking the $max(\cdot)$ over the collection of per bone indicators $\mathcal{O}_b(\cdot)$ in columns 3-13.

wise rigid composition of b elements; e.g. [27, 30]:

$$\mathcal{O}(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\mathcal{O}_b(\mathbf{x}|\boldsymbol{\theta})\} \tag{3}$$

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:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) = \max_b \{\bar{\mathcal{O}}^b_\omega(\mathbf{B}_b^{-1}\mathbf{x})\} \tag{4}$$

where and similar (2) 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 (2) to the one in (4), hence allowing each of the elements to be *adjusted* in shape conditional on the pose of the model:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) \approx \max_b \{\bar{\mathcal{O}}^b_\omega(\mathbf{B}_b^{-1}\mathbf{x}|\boldsymbol{\theta})\} \tag{5}$$

Similarly to (4) we use a *collection* of learnable indicator functions in rest pose $\{\bar{\mathcal{O}}^b_\omega\}$, and to encode pose conditionals we take inspiration from (2). More specifically, we express our model as:

$$\mathcal{O}_\omega(\mathbf{x}|\boldsymbol{\theta}) \approx \max_b \{\bar{\mathcal{O}}^b_\omega(\mathbf{B}_b^{-1}\mathbf{x}, \{\mathbf{B}_b^{-1}\mathbf{t}_0\})\} \tag{6}$$

Here $\mathbf{t}_0$ is the translation vector of the root bone in homogeneous coordinates, and the pose of the model is represented as $\{\mathbf{B}_b^{-1}\mathbf{t}_0\}$. Similarly to (4), we model this function via dense layers $\bar{\mathcal{O}}^b_\omega(.)$=MLP$^b_\omega(.)$. While the input dimensionality of this network is $3{+}B{\times}3$, which is similar to the dimensionality in (2), we will see that the necessary network capacity $|\omega|$ to achieve comparable approximation performance, especially in extrapolation settings, is much lower.

4.4. Technical details

We now detail the auxiliary losses we employ to facilitate learning, and the architecture of the network backbones.

Auxiliary loss – skinning weights. As most deformable models are equipped with skinning weights, we exploit this information to facilitate learning of the part-based models (i.e. "R" and "D"). In particular, 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:

$$\mathcal{L}_{\text{weights}}(\omega) = \tfrac{1}{V}\tfrac{1}{B}\sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\sum_{\mathbf{v}}\sum_b \left(\bar{\mathcal{O}}^b_\omega(\mathbf{v}|\boldsymbol{\theta}) - \mathcal{I}_b(\mathbf{v})\right)^2 \tag{7}$$

where $\mathcal{I}_b(\mathbf{v}){=}½$ when $b{=}b^*$, and $\mathcal{I}_b(\mathbf{v}){=}0$ otherwise – by convention, the ½ level set of the indicator is the surface our occupancy represents. In the supplementary material, we conduct an ablation study on the effectiveness of $\mathcal{L}_{\text{weights}}$ showing that this loss is *necessary* for effective shape decomposition. 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.

Auxiliary loss – parsimony. As parts create a whole via a simple union, nothing prevents unnecessary overlaps between parts. To remove this null-space from our training, we seek a minimal description by penalizing the *volume* of each part:

$$\mathcal{L}_{\text{parsimony}}(\omega) = \tfrac{1}{B}\sum_b \mathbb{E}_{\bar{\mathbf{x}}\in\mathbb{R}^3}\left[\|\bar{\mathcal{O}}^b_\omega(\bar{\mathbf{x}})\|^2\right] \tag{8}$$

This loss improves generalization, as quantified in the supplementary material.

Training. Given $\lambda_1{=}5e^{-1}$, and $\lambda_2{=}5e^{-3}$ as found through hyper-parameter tuning, the overall loss for our model is:

$$\mathcal{L}(\omega) = \mathcal{L}_{\text{occupancy}}{+}\lambda_1\mathcal{L}_{\text{weights}}{+}\lambda_2\mathcal{L}_{\text{parsimony}}$$

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

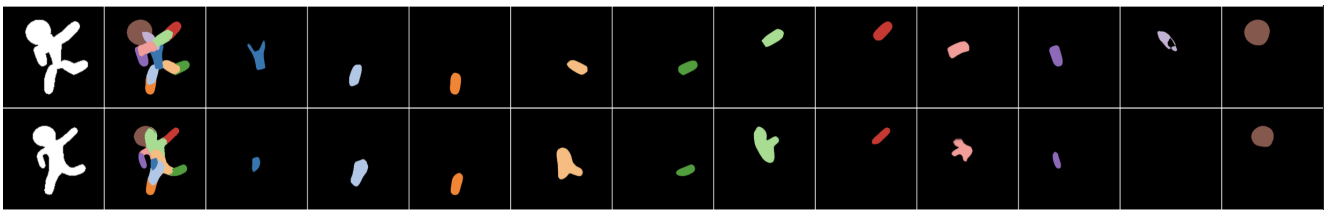

Figure 5: **Piecewise deformable model (D) –** is capable to deal with rigid (top), as well as non-rigid (bottom) deformations.

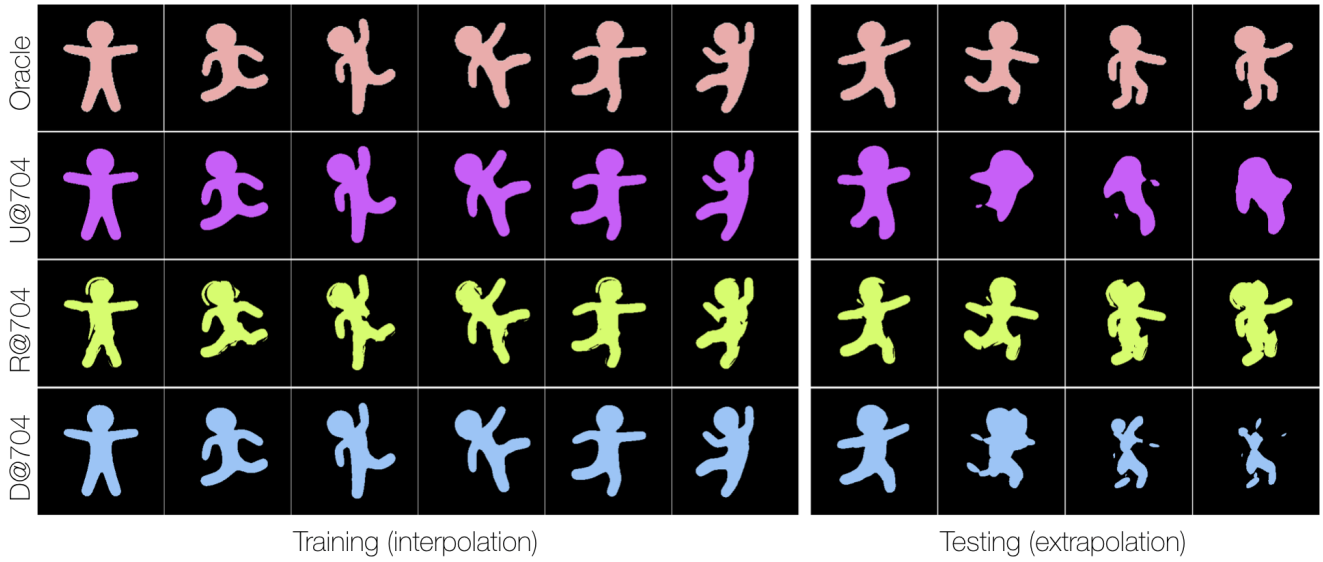

Figure 6: **Qualitative / 2D+T –** The animation is split into seen (training) and unseen (testing) poses.

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. For all the 2D experiments, we train the model for 120K iterations which takes approximately 5 hours on a single NVIDIA Tesla V100. For 3D experiments, the models are trained for 240K iterations for approximately 10 hours.

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 [29], but simplified to 4 layers. Each layer has a residual connection, and uses the Leaky ReLU activation function. All layers have the *same* size, which we vary from 88 to 760 according to the experiment (i.e., a backbone with 88 hidden units in the first layer will be marked as "@88"). For the piecewise (4) and deformable (6) models note the neurons are distributed across B different channels; e.g. with R@960 we mean that *each* of the B=24 branches will be processed by dense layers having $960/24$=40 neurons. Similarly to the use of grouped filters/convolutions [13, 19], note that such a structure allows for significant performance boosts compared to unstructured models (2), as the different branches can be executed in parallel on separate compute devices.

5. Evaluation

We employ two datasets to evaluate our method in 2D and 3D. The datasets consist of a rest configuration surface, sampled indicator functions values, bone transformation frames per pose, and skinning weights. The ground truth indicator functions were robustly computed via generalized winding numbers [15], and are evaluated in a regular grid surrounding the deformed surface with additional samples on the surface. The performance of the models can be evaluated by comparing the Intersection over Union (IOU) of the predicted indicator values against the ground truth samples on a regular grid.

5.1. Analysis on 2D data

Our gingerbread dataset consists of 100 different poses sampled from a temporally coherent animation. The animation drives the geometry in two different ways: ① in the rigid dataset, we have a collection of surfaces, and each surface region is rigidly attached to a single bone which *does not* change shape as the pose changes; ② in the blended dataset, we employ the skinning weights to deform the surfaces via LBS. Our 2D results are summarized in Figure 3: given enough neural capacity, both the unstructured and

Training	Testing	U@192	R@192	D@192	U@960	R@960	D@960
Jumping Jacks	Jumping Jacks	.92	.94	**.95**	.96	.96	**.97**
Punching	Punching	.93	**.95**	.94	.97	.96	**.97**
Running on Spot	Running on Spot	.92	**.94**	.94	.97	.95	**.97**
One Leg Jump	One Leg Jump	.92	**.94**	.94	.96	.96	**.97**
mean	(interp.)	.92	**.95**	.94	.97	.96	**.97**

interpolation

Training	Testing	U@192	R@192	D@192	U@960	R@960	D@960
!Jumping Jacks	Jumping Jacks	.21	**.93**	.53	.34	**.93**	.71
!Punching	Punching	.66	.92	**.93**	.72	**.95**	.94
!Running on Spot	Running on Spot	.71	**.94**	**.94**	.76	**.95**	.96
!One Leg Jump	One Leg Jump	.68	.94	**.94**	.69	**.95**	.96
mean	(extrap.)	.56	**.94**	.83	.63	**.94**	.89

extrapolation

Table 1: **Quantitative / 3D –** Mean IoU across baselines on 3D sequences from the AMASS dataset [26]. The top part of the table tests *interpolation*, while the second part *extrapolation* (leave one out) performance of our neural shape approximation. These results can be better appreciated qualitatively in *supplementary material* video.

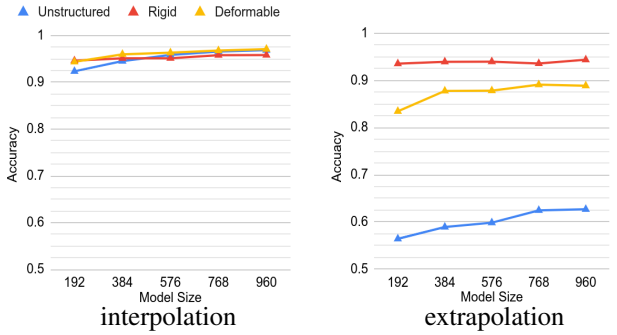

Figure 7: We evaluate the average performance (IoU) of the various models as we sweep the complexity of the network in the range $\{192 \cdots 960\}$. We report the results in both interpolation (left) and extrapolation (right) regimes.

deformable model are able to overfit to the training data. Note that since the animation produced via skinning exhibits *highly* non-rigid (i.e. blended) deformations, the rigid model struggles.

Unstructured model – "U". Looking at overfitting results can be misleading, and, in this sense, the fundamental limitations of the unstructured model are revealed in Figure 6. The performance of the unstructured model gives reasonable reconstruction across poses seen within the training set, but struggles to generalize to new poses – the more different the pose is from those in the training set, the worse the IoU score.

Piecewise rigid model – "R". Training the representation in (4) via SGD is effective *when* the data can truly be modeled by a piecewise rigid decomposition; see Figure 4 (top). When the same network is trained on a dataset which violates this assumption, the learning performance degrades significantly; see Figure 4 (bottom). The rigid animation is recreated exactly, but the blended animation has incorrect *blurred* boundaries, and is missing portions of the bone indicators. Note that adding more capacity to the network brought no further improvements in performance, as the dataset violates the core assumption of the rigid model.

Piecewise deformable model – "D". Skinned deformation models give smooth transitions between bones, making a single continuous surface across the range of deformations. Conditioning the network with pose as in (6) allows the network to learn the relative deformation of the parts across poses. When the surface cannot be simply modelled with a piecewise rigid decomposition, the piecewise deformable model performs significantly better. This improvement can clearly be seen by comparing the results of Figure 5 to those of Figure 4. While in interpolation scenarios (re-playing the frames of a known animation) the deformable model performs excellently, it struggles when dealing with extrapolation (Figure 6). The extrapolation performance on a realistic 3D dataset (Section 5.2) is better, perhaps because physically correct deformations are not as exaggerated as in our gingerbread example, hence more predictable.

5.2. Analysis on 3D data

The AMASS dataset [26] is a large-scale collection of 3D human motion driven by SMPL [23]. In this paper, we use the "DFaust_67" subset of AMASS which has 10 humanoid characters performing different motions. We select the "50002" subject (see Figure 1) and consider four different sequences. The deformation model of the dataset involves LBS with pose-space correctives [23]. As each sequence contains 200 to 400 frames, the overall training dataset contains ≈1000 3D objects, which is roughly the same as the biggest classes (planes, chairs) of the ShapeNet dataset [3]. Note that the model is trained on a *single* character and is *not* expected to generalize across characters, but rather across *animation* sequences. Further, note that we did not balance the sampling density to focus the network training on small features such as fingers and face details, as these are not animated in AMASS.

Analysis. As visualized in Table 1 and Figure 7, the deformable model is able to achieve high IOU scores with fewer model parameters than would be required with a fully unstructured network. On the AMASS dataset, the rigid model performed well on interpolation tasks, and did not suffer failures analogous to those found in 2D; see Figure 6 – we believe this is due to the fact that our gingerbread character presents non-rigid deformations *far beyond* those the level that are present in the AMASS dataset. Note how both rigid and deformable models are *significantly* better than the unstructured baseline in generalizing to unseen

poses; see qualitative results in Figure 1, as well as in the supplementary material. Note how the plot in Figure 7 reveals how the rigid model is able to extrapolate much better than the unstructured model. Nonetheless, the deformable model results are still not optimal as the model was unable to properly extrapolate the pose-dependent correctives for unseen poses.

6. Conclusions

We introduced the problem of geometric modeling of deformable (solid) models from a neural perspective. We showed how unstructured baselines require a significantly larger neural budget compared to structured baselines, but more significantly, they simply fail to generalize. Amongst structured baselines the deformable models performs best at interpolation, while the rigid model leads the extrapolation benchmarks. It would be interesting to understand how to combine these two models and inherit *both* behaviors. Note that the deformable model ("D") is still usable in applications as far as the query poses are sufficiently similar to those seen at training time.

Application. Our approach can be applied to a number of problems. These include representation of complex articulated bodies such as human characters, object intersection queries for computer vision registration and tracking, collision detection for computer games and other applications, and compression of mesh sequences. In all these applications neural shape approximation allows different trade-offs of *efficiency vs. detail* to be handled using the same general approach.

Future directions. One natural direction for future work would be to reduce the amount of supervision needed. To name a few goals in increasing order of complexity: ① Can we learn the posing transformations $\{\tilde{\mathbf{B}}_b\}$ and perhaps also the rest transformations $\{\bar{\mathbf{B}}_b\}$ automatically? ② Can the representation be generalized to capture *collections* of deformable bodies? (i.e. the $\boldsymbol{\beta}$ parameters of SMPL [23]). ③ Can the signed distance function, rather than occupancy be learnt as well? ④ Is NASA a representation suitable to differentiable rendering? ⑤ Can a 3D representation of articulated motion be learnt from 2D supervision alone?

7. Acknowledgements

We would like to particularly thank Paul Lalonde for the initial project design, and Gerard Pons-Moll for his help accessing the AMASS data. We would also like to thank David I.W. Levin, Alec Jacobson, Hugues Hoppe, Nicholas Vining, Yaron Lipman, and Angjoo Kanazawa for the insightful discussions.

References

[1] Dragomir Anguelov, Daphne Koller, Hoi-Cheung Pang, Praveen Srinivasan, and Sebastian Thrun. Recovering articulated object models from 3d range data. In *Uncertainty in Artificial Intelligence*, 2004. 2

[2] Stephen W. Bailey, Dave Otte, Paul Dilorenzo, and James F. O'Brien. Fast and deep deformation approximations. *SIGGRAPH*, 2018. 1, 2

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015. 6

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *CVPR*, 2019. 1, 2, 3

[5] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *ICCV*, 2019. 2

[6] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *CVPR*, 2019. 1

[7] Fernando de Goes, Siome Goldenstein, and Luiz Velho. A hierarchical segmentation of articulated bodies. In *SGP*, 2008. 2

[8] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *arXiv:1909.05736*, 2019. 1, 2

[9] Boyang Deng, Simon Kornblith, and Geoffrey Hinton. Cerberus: A multi-headed derenderer. *arXiv:1905.11940*, 2019.

[10] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: deep generative network for structured deformable mesh. *ACM TOG*, 2019. 2

[11] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-m\^ ach\'e approach to learning 3d surface generation. *arXiv preprint arXiv:1802.05384*, 2018. 1

[12] Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. *ACM TOG*, 2011. 2

[13] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep Roots: Improving CNN efficiency with hierarchical filter groups. In *CVPR*, 2017. 5

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1

[15] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM TOG*, 2013. 2, 3, 5

[16] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J.P. Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH Courses*, 2014. 2

[17] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *SIGGRAPH*, 2005. 2

[18] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 1

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 5

[20] Binh Huy Le and Zhigang Deng. Smooth skinning decomposition with rigid bones. *ACM TOG*, 2012. 2

[21] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 2

[22] Ming C. Lin, USA Dinesh Manocha, and Jon Cohen. Collision detection: Algorithms and applications, 1996. 2

[23] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *SIGGRAPH Asia*, 2015. 2, 3, 6, 7

[24] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987. 1

[25] Dominik Lorenz, Leonard Bereska, Timo Milbich, and BjÃűrn Ommer. Unsupervised part-based disentangling of object shape and appearance. *arXiv:1903.06946*, 2019. 2

[26] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. *ICCV*, 2019. 6

[27] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Graphics Interface*, 2013. 4

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *arXiv:1812.03828*, 2018. 1, 2, 3

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. *CVPR*, 2019. 1, 2, 3, 5

[30] Edoardo Remelli, Anastasia Tkach, Andrea Tagliasacchi, and Mark Pauly. Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In *ICCV*, 2017. 2, 4

[31] Hanan Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN 0-201-50300-X. 2

[32] Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *CVPR*, 2017. 1

[33] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *arXiv:1904.12356*, 2019. 1

[34] Julien Valentin, Cem Keskin, Pavel Pidlypenskyi, Ameesh Makadia, Avneesh Sud, and Sofien Bouaziz. Tensorflow graphics: Computer graphics meets deep learning. https://github.com/tensorflow/graphics, 2019. 1