Supplementary Material for Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks.

Despoina Paschalidou[1,5,6] Angelos Katharopoulos[3,4] Andreas Geiger[1,2,5] Sanja Fidler[6,7,8]
[1]Max Planck Institute for Intelligent Systems Tübingen [2]University of Tübingen
[3]Idiap Research Institute, Switzerland [4]École Polytechique Fédérale de Lausanne (EPFL)
[5]Max Planck ETH Center for Learning Systems [6]NVIDIA [7]University of Toronto [8]Vector Institute
{firstname.lastname}@tue.mpg.de angelos.katharopoulos@idiap.ch sfidler@nvidia.com

Abstract

*In this **supplementary document**, we first present a detailed overview of our network architecture and the training procedure. We then provide ablations on how different components of our system impact the performance of our model on the single-view 3D reconstruction task. Subsequently, we provide additional experimental results on the D-FAUST [2], the FreiHAND [21] and ShapeNet [3] dataset. Next, we provide additional evidence for the representation consistency of our method.*

1. Implementation Details

In this section, we provide a detailed description of our network architecture. We then describe our training protocol, our sampling strategy and provide details on the computation of the metrics during training and testing. Finally, we also provide additional details regarding our baselines.

1.1. Network Architecture

Here we describe the architecture of each individual component of our model (from Fig. 2 in the main submission). Our architecture comprises two main components: (i) a *feature extractor* that maps the input into a vector of per-primitive shape embeddings $\{\mathbf{C}_m\}_{m=1}^M$ and (ii) the *conditional homeomorphism* that learns a family of homeomorphic mappings between the sphere and the target shape, conditioned on the shape embedding.

Feature Extractor: The first part of the *feature extractor* is the feature encoder. In our experiments, the feature encoder is implemented with a ResNet-18 architecture [11] that is pre-trained on ImageNet [8]. From the original architecture, we remove the final fully connected layer and keep only the feature vector of length 512 after average pooling. Subsequently, we use 2 fully connected layers to map this to the global feature vector $\mathbf{F} \in \mathbb{R}^{256}$. During training, we use the pre-trained batch statistics for normalization. The primitive embedding $\{\mathbf{P}_m\}_{m=1}^M$ is a matrix of size $M \times 256$, that stores a vector per-primitive index that is concatenated with the global image feature $\mathbf{F}$ and outputs a vector of shape embeddings $\{\mathbf{C}_m \in \mathbb{R}^{512}\}_{m=1}^M$ for every primitive. A pictorial representation of the feature extractor is provided in Fig. 1.

Conditional Homeomorphism: The conditional homeomorphism is implemented with a Real NVP [9] augmented as described in the main submission. It comprises 4 coupling layers and each coupling layer consists of three components: $s_{\boldsymbol{\theta}}(\cdot)$ that predicts the **scaling factor** s, $t_{\boldsymbol{\theta}}(\cdot)$ that predicts the **translation amount** t and $p_{\boldsymbol{\theta}}(\cdot)$ that maps the input 3D point to a higher dimensional space before concatenating it with the per-primitive shape embedding $\mathbf{C}_m$. In particular, $s_{\boldsymbol{\theta}}(\cdot)$ is a 3-layer MLP with hidden size equal to 256. After each layer we add ReLU non-linearities except for the last layer, where we use a hard tanh with thresholds -10 and 10 (see Fig. 2a). Note that we use the tanh to avoid numerical instabilities when computing the exponential of the scaling factor. $t_{\boldsymbol{\theta}}(\cdot)$ is implemented with the same architecture apart from the tanh activation in the final layer (see Fig. 2b). Finally, $p_{\boldsymbol{\theta}}(\cdot)$ is implemented with a 2-layer MLP with ReLU non-linearities with hidden size 256

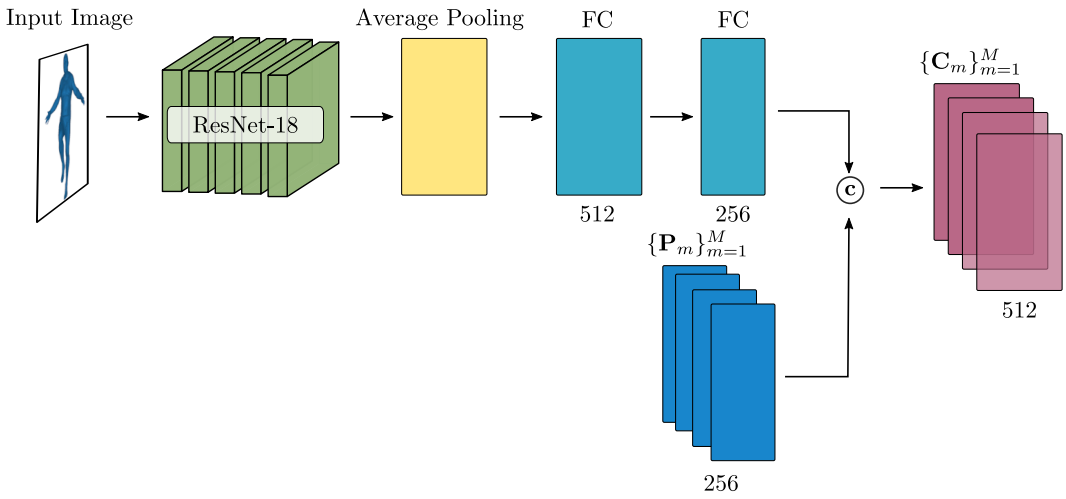
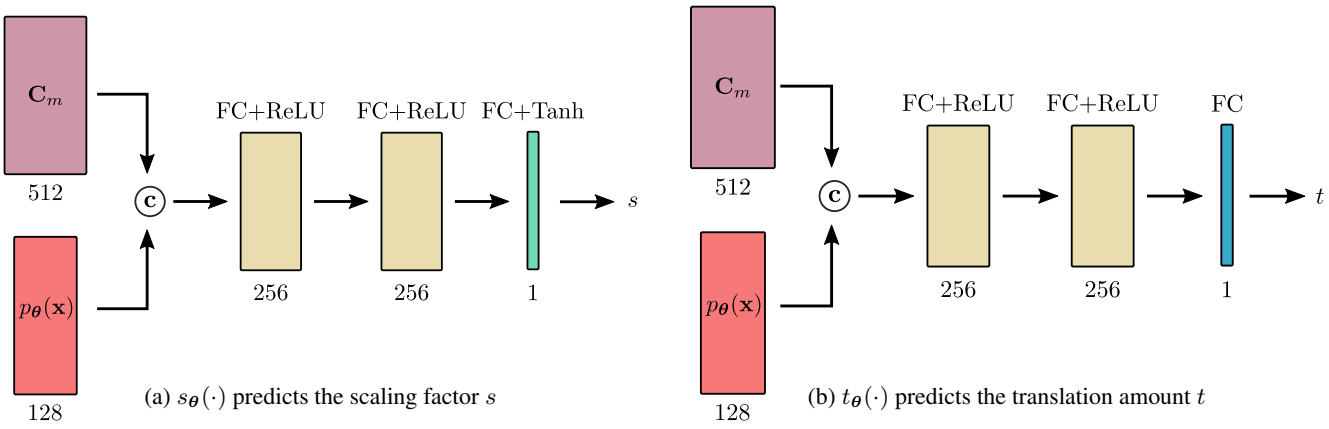


Figure 1: **Feature Extractor.**. The feature extractor predicts a vector of per-primitive shape embeddings $\{\mathbf{C}_m\}_{m=1}^M$ conditioned on the input image. Note that depending on the type of the input (e.g. pointcloud, voxel grid), a different feature encoder module needs to be employed.



(a) $s_{\boldsymbol{\theta}}(\cdot)$ predicts the scaling factor s

(b) $t_{\boldsymbol{\theta}}(\cdot)$ predicts the translation amount t

(c) $p_{\boldsymbol{\theta}}(\cdot)$ maps the input point (x_i, y_i, z_i) into a higher dimensional feature space

Figure 2: **Conditional Homeomorphism.** The conditional homeomorphism is the backbone of our architecture that given a vector of per-primitive shape embeddings and a set of points on the sphere in the latent space maps them in the primitive space of the target object and vice versa. Here, we visualize $s_{\boldsymbol{\theta}}(\cdot)$, $t_{\boldsymbol{\theta}}(\cdot)$ and $p_{\boldsymbol{\theta}}(\cdot)$ that comprise our *affine coupling layer*.

and output size 128. An illustration of the components of the conditional homeomorphism is provided in Fig. 2. As discussed in the main submission, we split the dimensions of the input point (x_i, y_i, z_i) in order to scale and translate one of them based on the other two. In practice, this is efficiently implemented via masking one of the dimensions, as illustrated in Fig. 2c with gray. For completeness, in equation 1 we provide the inverse of the conditional affine coupling layer from the main paper,

$$\begin{aligned} x_i &= x_o \\ y_i &= y_o \\ z_i &= \left(z_o - t_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_o, y_o\right)\right]\right)\right) \exp\left(-s_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_o, y_o\right)\right]\right)\right) \end{aligned} \tag{1}$$

1.2. Training Protocol

In all our experiments, we use the Adam optimizer [12] with learning rate $\eta = 10^{-4}$ and no weight decay. For the other hyperparameters of Adam we use the PyTorch defaults. Depending on the number of primitives, we train our model with a different batch size and a different number of iterations. In particular, for 2 primitives, we use a batch size of 6 for 100k iterations and we aggregate the gradients over 2 batches. For the experiments, with 5 primitives, we use a batch size of 4 for 150k iterations and we again aggregate the gradients over 2 batches. For 8 and 10 primitives, we use a batch size of 4 for 200k iterations and aggregate the gradients over 4 batches. We do not perform any data augmentation.

We weigh the loss terms of Eq. 11 in our main submission with $1.0, 0.1, 0.01, 0.1, 0.01$. Note that we use smaller weights for $\mathcal{L}_{norm}$ and $\mathcal{L}_{cover}$ since they act as regularizers and we want our model to focus primarily on learning geometrically accurate and semantically meaningful primitives. The impact of each term is discussed in detail in Sec. 2.3. The temperature parameter τ in the occupancy $\mathcal{L}_{occ}$ and the overlap loss $\mathcal{L}_{overlap}$ is set to 4×10^{-3}. In addition, the k term for the $\mathcal{L}_{overlap}$ is set to 1.95 since we want to penalize overlapping primitives but ensure that they will be connected. Note that $k = 1$ results in disconnected primitives and $k = 2$ allows primitives to overlap in pairs of two. Hence we select $k = 1.95$ as we empirically observe that it balances connectivity and interpenetration. Finally, we set the k to be equal to 10 for the $\mathcal{L}_{cover}$, as we empirically observe that it leads to good performance.

1.3. Sampling Strategy

In this section, we discuss our sampling strategy for generating the surface samples $\mathcal{X}_t = \{\mathbf{x}_i, \mathbf{n}_i\}_{i=1}^{N}$, the occupancy pairs $\mathcal{X}_o = \{\mathbf{x}_i, o_i\}_{i=1}^{V}$ and the points on the sphere surface $\mathcal{Y}_s = \{\mathbf{y}_j\}_{j=1}^{K}$.

Surface Samples: We generate samples on the surface of the target mesh by sampling a face with probability proportional to its area and then sampling a point uniformly in that face. We use the corresponding face normals to generate point-normal pairs from the target mesh. During training, we randomly sample $2,000$ points and normals on the target mesh.

Occupancy Pairs: We generate occupancy pairs by sampling $100,000$ points uniformly in the unit cube centered at $(0, 0, 0)$ for each mesh. Subsequently, we compute which of these points lie inside or outside the mesh to generate the occupancy labels. However, because the target objects have a small volume compared to the unit cube, resampling uniformly from the $100,000$ occupancy pairs results in a small amount of points with positive labels. This yields bad reconstructions for parts with small volume such as hands and fingers. To address this, we sample from an unbalanced distribution that, in expectation, results in an equal number of points with positive and negative labels. However, we also compute importance sampling weights in order to reweigh our loss and create an unbiased estimator of the loss with uniform sampling similar to [15]. During training, we sample $5,000$ occupancy pairs.

Sphere Points: In order to generate points on the surface of each primitive, we sample points uniformly on the surface of a sphere with radius r by sampling points from a 3-D isotropic Gaussian and normalizing their length to r. Note that this does not generate uniform samples on the surface of the primitive due to different amounts of stretching and contracting of the sphere by each homeomorphism. However, we empirically observe that this sampling does not impact negatively the training of our model, thus we leave sampling uniformly on the surface of each primitive for future work. During training we sample 200 points for each primitive. In addition, to generate meshes for the predicted primitives, we also need to compute sphere surface points and corresponding faces. We achieve this using the *UV* parametrization of the sphere [5]. For visualization purposes we use $2,000$ points sampled uniformly in latitude and longitude.

1.4. Metrics

As mentioned in the main submission, we evaluate our model and our baselines using the volumetric Intersection-over-Union (IoU) and the Chamfer-L_1 distance. We obtain unbiased low-variance estimates for the IoU by sampling $100,000$ points from the bounding volume and determining if the points lie inside or outside the target/predicted mesh. Similarly, we obtain an unbiased low-variance estimator of the Chamfer-L_1 distance by sampling $10,000$ points on the surface of the target/predicted mesh.

Volumetric IoU is defined as the quotient of the volume of the intersection of the target O_t and the predicted O_p object and the volume of their union.

$$\text{IoU}(O_t, O_p) = \frac{|\, V(O_t \cap O_p)\,|}{|\, V(O_t \cup O_p)\,|} \tag{2}$$

where $V(.)$ is a function that computes the volume of a mesh.

The Chamfer-L_1 distance is defined between a set of points sampled on the surface of the target and the predicted mesh. We denote $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ the set of points sampled on the surface of the target mesh and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^M$ the set of points sampled on the surface of the predicted mesh.

$$\text{Chamfer-}L_1(\mathcal{X}, \mathcal{Y}) = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{x} - \mathbf{y}\| + \frac{1}{M} \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\| \tag{3}$$

The first term of (3) measures the *completeness* of the predicted shape, namely how far is on average the closest predicted point from a ground-truth point. The second term measures the *accuracy* of the predicted shape, namely how far on average is the closest ground-truth point from a predicted point. Note that during the evaluation we do not sample points on the sphere to generate points on the surface of the primitives. Instead we use the predicted meshes to ensure that the generated points are uniformly distributed on the surface of the predicted object.

1.5. Baselines

In this section, we provide additional details regarding our baselines. For a fair comparison, all baselines use the same feature extractor network, namely ResNet18 [11].

SQs: In SQs [15], the authors employ a convolutional neural network (CNN) to regress the parameters of a set of superquadric surfaces that best describe the target object, in an unsupervised manner, by minimizing the Chamfer-distance between points on the target and the predicted shape. In order to have predictions, with variable number of primitives, each primitive is associated with an existence probability. When the existence probability of a primitive is below a threshold this primitive is not part of the assembled object. We train SQs [15][1] using the provided PyTorch [17] implementation with the default parameters. Similar to our model, we sample 200 points on the surface of each primitive and $2,000$ on the surface of the target object. We train SQs with a batch size of 32 until convergence.

H-SQs: H-SQs [16] consider a hierarchical, geometrically more accurate decomposition of parts using superquadrics. In particular, H-SQs employ a neural network architecture that recursively partitions objects into their constituent parts by building a latent vector that jointly encodes both the part-level hierarchy and the part geometries. H-SQs are trained in an unsupervised fashion, with an occupancy loss function such that the model learns to classify whether points sampled in the bounding box that contains the target object lie insider or outside the union of the predicted primitives. We train H-SQs [16] using the PyTorch code provided by the authors. Note that the maximum number of primitives for H-SQs needs to be a power of 2, hence we train with a different number of primitives than the other baselines. Again, we train H-SQs with a batch size of 32 until convergence.

CvxNet: CvxNet [7] propose to reconstruct a 3D shape as a collection of convex hulls. We train CvxNet [7][2] using the provided TensorFlow [1] code with the default parameters. To ensure that the evaluation is consistent and fair, we extract meshes using their codebase and evaluate the predicted primitives using our evaluation code. For all datasets, we train CvxNet with a batch size of 32 until convergence using their default parameters.

OccNet: In our evaluation, we also include OccNet [14][3] which we train using the provided code by the authors. In particular, OccNet is trained with a batch size of 32 until convergence. While OccNet does not consider any part-based decomposition of the target object and is not directly comparable with our model, we include it in our analysis as a typical representative of powerful implicit shape extraction techniques

[1] https://superquadrics.com/
[2] https://cvxnet.github.io/
[3] https://github.com/autonomousvision/occupancy_networks

2. Ablation Study

In this section, we investigate how various components of our system affect the performance of Neural Parts on the single-view 3D reconstruction task on the D-FAUST dataset. In Sec. 2.1, we examine the impact of the INN on the performance of our model. Next, in Sec. 2.2, we investigate the impact of $p_{\boldsymbol{\theta}}(\cdot)$ and in Sec. 2.3, we discuss the effect of each loss term on the overall performance of our model. Unless stated otherwise, we train all models with 5 primitives for 300 epochs.

2.1. Invertibility

In this section, we examine the impact of the Invertible Neural Network (INN) on the performance of Neural Parts. To this end, we implement a variant of our model that does not consider the inverse mapping of the homeomorphism, namely without the $\phi_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$. As a result, for this variant it is not possible to compute the union of parts, as formulated in Eq. 10 in our main submission, namely we cannot discard points on a primitive's surface that are internal to another primitive when generating points on the surface of the predicted shape. Moreover, we cannot impose any additional constraints on the predicted primitives. Therefore, we train this variant using the reconstruction loss between the target $\mathcal{X}_t$ and the predicted $\mathcal{X}_{\hat{p}}$ shape, $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_{\hat{p}})$ from Eq 12 in our main submission as follows:

$$\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_{\hat{p}}) = \frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_i \in \mathcal{X}_t} \min_{\mathbf{x}_j \in \mathcal{X}_{\hat{p}}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 + \frac{1}{|\mathcal{X}_{\hat{p}}|} \sum_{\mathbf{x}_j \in \mathcal{X}_{\hat{p}}} \min_{\mathbf{x}_i \in \mathcal{X}_t} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{4}$$

where $\mathcal{X}_{\hat{p}} = \mathbf{x} \in \bigcup_m \mathcal{X}_p^m$, namely the union of surface points on each primitive.

Furthermore, we introduce an additional baseline that replaces the INN with an MLP. Similar to AtlasNet [10], we use a 4-layer MLP to learn the homeomorphic mappings between the sphere and the target object. Since the homeomorphism is implemented via an MLP it is not possible to define its inverse mapping. Thus, also this baseline is trained using the loss of (4). Note that this baseline is slightly different from AtlasNet [10] because instead of having a single MLP for each homeomorphism, we have one MLP for the M homeomorphisms. In particular, for this baseline, we implement the decoder architecture proposed in [10].

	w/o $\phi_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$	AtlasNet - sphere	Ours
IoU	0.639	$*$	0.673
Chamfer-L_1	0.119	0.087	0.097

Table 1: **Ablation Study on INN.** This table shows a quantitative comparison of our approach wrt. a variant of our model that does not consider the inverse mapping of the homeomorphism and a second baseline that replaces the INN with an MLP for learning the homeomorphic mapping. We refer to the latter as AtlasNet - sphere, as it is a variant of the original AtlasNet method [10]. We note that our model outperforms both baseline both in terms of IoU (↑) and Chamfer-L_1 distance (↓).

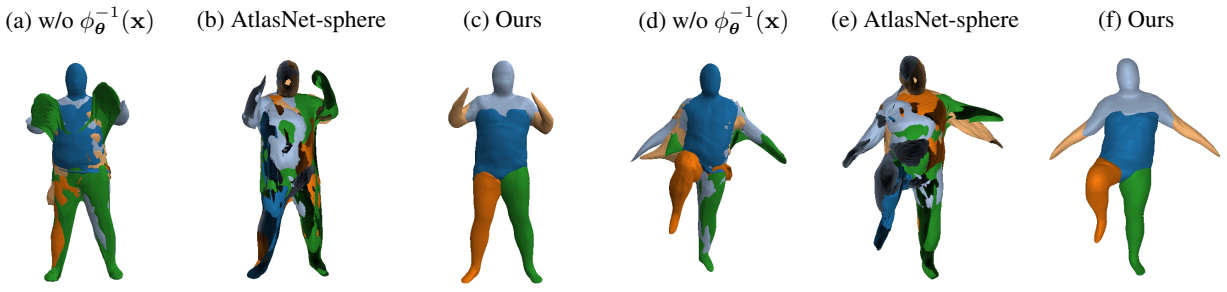


Figure 3: **Qualitative evaluation on the impact of the INN.** We visualize the predicted primitives for our model (third and sixth column), a variant of our model that ignores the inverse mapping of the homeomorphism (first and fourth column) and the AtlasNet-sphere baseline that implements the homeomorphism with an MLP.

Tab. 1 summarizes the quantitative comparison between our model and the aforementioned baselines. Note that for the AtlasNet-sphere baseline, we cannot compute the IoU since it is not possible to enforce that the predicted primitives do

not degenerate to "sheets", which contain no points (zero volume primitives). Indeed, from our evaluation, we observe that almost all predicted primitives (see Fig. 4) have zero volume and also inverted normals, which means that the sphere has folded on itself (self-intersections). This is only possible if the MLP is not implementing a homeomorphism. Due to the aforementioned issues, the AtlasNet-sphere cannot be used for learning primitives, however, we validate that it achieves high scores in terms of Chamfer-L_1 distance, which is justified as AtlasNet-sphere is optimized solely on this metric.

On the contrary, our method always learns valid homeomorphisms even when the inverse mapping ($\phi_{\boldsymbol{\theta}}^{-1}(\cdot)$) is not used during training. This stems from the use of the invertible network, which implements a bijection that makes self-intersections impossible. Formally, for a self-intersection, two different points on the sphere need to be mapped onto the same point on the primitive space. However, this contradicts the bijection; hence our model cannot have self-intersections. This becomes evident from Fig. 3a+Fig. 3d, where we observe that the predicted primitives do not have inverted normals. Note that without the inverse mapping we cannot enforce that primitives will not interpenetrate, hence they are not semantically meaningful.

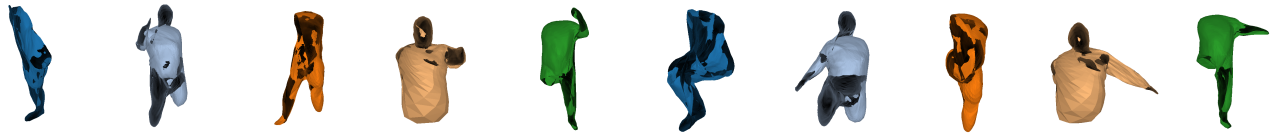


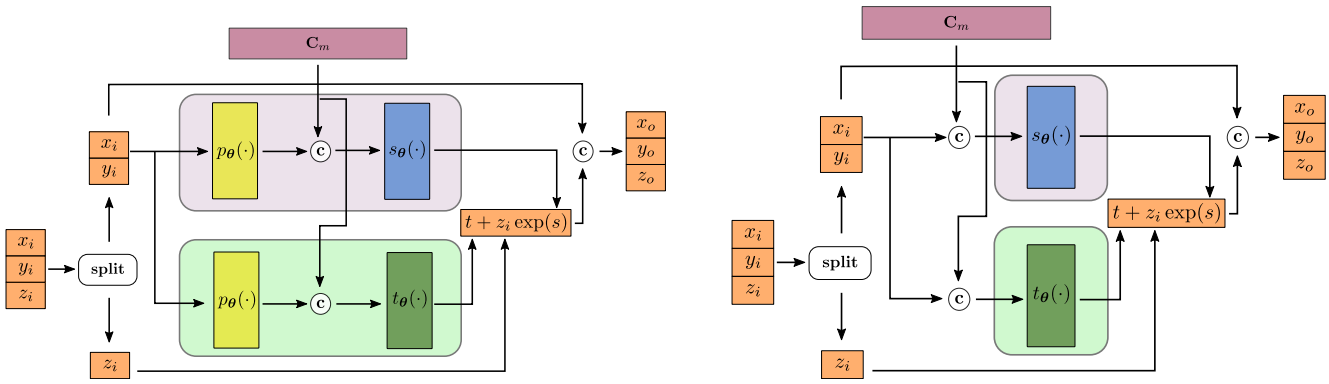
Figure 4: **Predicted primitives of AtlasNet-sphere.** We visualize the individual primitives for the predictions of AtlasNet-sphere. Note that some of the predicted primitives are hollow (first, third, sixth and eighth), some have holes (second, seventh) and all of them have inverted normals.

2.2. Effect of $p_{\boldsymbol{\theta}}(\cdot)$

The original Real NVP [9] cannot be directly applied in our setting as it does not consider a shape embedding. To address this, we augment the affine coupling layer as follows: we first map (x_i, y_i) into a higher dimensional feature vector using a mapping, $p_{\boldsymbol{\theta}}(\cdot)$, implemented as an MLP. This is done to increase the relative importance of the input point before concatenating it with the high-dimensional shape embedding $\mathbf{C}_m$. The *conditional affine coupling layer* becomes

$$\begin{aligned} x_o &= x_i \\ y_o &= y_i \\ z_o &= z_i \exp\left(s_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right)\right) + t_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right) \end{aligned} \tag{5}$$

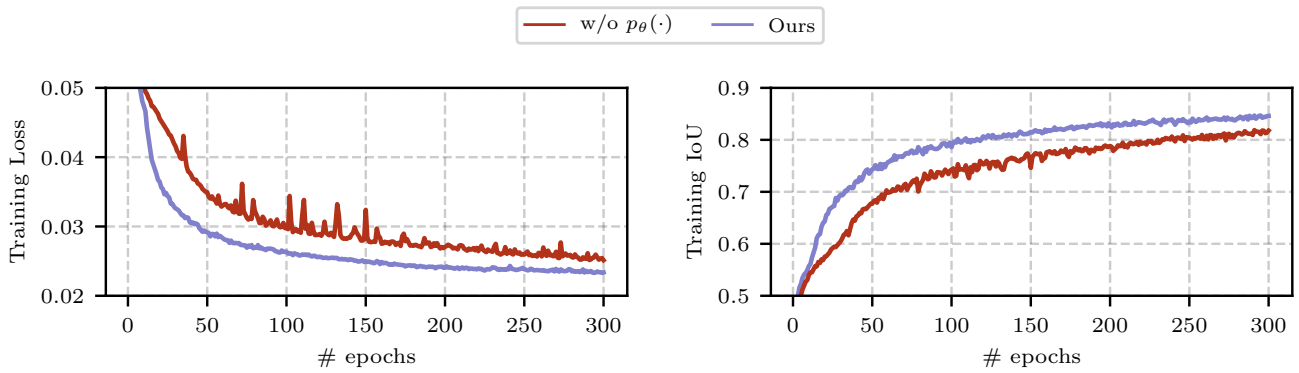
where $[\cdot;\cdot]$ denotes concatenation. A graphical representation of our conditional coupling layer is provided in Fig. 5a. In this section, we examine the impact of the mapping $p_{\boldsymbol{\theta}}(\cdot)$ on the performance of our model. In particular, instead of first mapping the input point (x_i, y_i, z_i) into a higher dimensional space and then concatenating it with the per-primitive shape embedding $\mathbf{C}_m$, we concatenate it as is (see Fig. 5b).



(a) Ours

(b) Conditional Coupling Layer w/o $p_{\boldsymbol{\theta}}(\cdot)$

Figure 5: **Conditional Coupling Layer.** Pictorial representation of our conditional affine coupling layer. The input point (x_i, y_i, z_i) is passed into a *coupling layer* that scales and translates one dimension of the input based on the other two and the per-primitive shape embedding $\mathbf{C}_m$. The scale factor s and the translation amount t are predicted by two MLPs, $s_{\boldsymbol{\theta}}(\cdot)$ and $t_{\boldsymbol{\theta}}(\cdot)$. $p_{\boldsymbol{\theta}}(\cdot)$ is another MLP that increases the dimensionality of the input point before it is concatenated with $\mathbf{C}_m$.



(a) Evolution of Training Loss

(b) Evolution of Training IoU

Figure 6: **Training Convergence w/o $p_{\boldsymbol{\theta}}(\cdot)$.** We illustrate the training evolution over 300 epochs in terms of training loss (Eq. 11 in the main submission) and IoU for our method with (**purple**) and without $p_{\boldsymbol{\theta}}(\cdot)$ (**red**). We observe that our method converges smoother and to a lower loss and higher IoU when using $p_{\boldsymbol{\theta}}(\cdot)$.

We observe that removing $p_{\boldsymbol{\theta}}(\cdot)$ makes it harder for the network to produce a scaling and translation dependent on the input point due to the large difference in the number of dimensions compared to the shape embedding $\mathbf{C}_m$. This results in slower convergence in comparison to our full model (see Fig. 6). Moreover, we also notice that the reconstruction quality becomes worse both in terms of IoU and Chamfer-L_1 distance (see Tab. 2).

	w/o $p_{\boldsymbol{\theta}}(\cdot)$	Ours
IoU	0.638	0.673
Chamfer-L_1	0.106	0.097

Table 2: **Ablation Study on $p_{\boldsymbol{\theta}}(\cdot)$.** This table shows a numeric comparison of our approach wrt. a variant of our model that does not utilize $p_{\boldsymbol{\theta}}(.)$, namely instead of first mapping the input point (x_i, y_i, z_i) into a higher dimensional space, we directly concatenate the input point with the per-primitive shape embedding $\mathbf{C}_m$.

2.3. Loss Functions

Neural Parts are trained in an unsupervised fashion, without any primitive annotations. To address, the lack of part-level supervision, we learn this task by minimizing the geometric distance between the target and the predicted shape. In particular, our optimization objective comprises five loss terms that enforce that the predicted primitives are geometrically accurate and semantically meaningful.

$$\mathcal{L} = \mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p) + \mathcal{L}_{occ}(\mathcal{X}_o) + \mathcal{L}_{norm}(\mathcal{X}_t) + \mathcal{L}_{overlap}(\mathcal{X}_o) + \mathcal{L}_{cover}(\mathcal{X}_o). \tag{6}$$

In this section, we discuss how each loss term affects the performance of our model on the single-view 3D reconstruction task. In particular, we train 5 variants of our model and for each one we omit one of the loss terms. We provide both quantitative (see Tab. 3) and qualitative comparison (see Fig. 7) for the 5 different scenarios. For a fair comparison all models are trained for the same number of epochs.

	w/o $\mathcal{L}_{occ}$	w/o $\mathcal{L}_{rec}$	w/o $\mathcal{L}_{norm}$	w/o $\mathcal{L}_{overlap}$	w/o $\mathcal{L}_{cover}$	Ours
IoU	0.642	0.643	0.669	0.670	0.668	0.673
Chamfer-L_1	0.125	0.150	0.096	0.098	0.096	0.097

Table 3: **Ablation Study on Loss Terms.** We investigate the impact of each loss term by training our model without each one of them. We report the IoU (↑) and the Chamfer-L_1 distance (↓) on the single-view 3D reconstruction task on D-FAUST dataset.

Figure 7: **Ablation Study on Loss Terms.** Qualitative evaluation of the impact of the 5 loss terms on the performance of our model. We train Neural Parts, without each loss term and visualize the predicted primitives.

w/o Reconstruction Loss: The reconstruction loss is a bidirectional Chamfer loss between the surface points on the target and the predicted shape. We note that removing $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p)$ results in degenerate primitive arrangements that fail to capture the object geometry.

w/o Occupancy Loss: The occupancy loss enforces that the free and the occupied space of the predicted and the target

object will coincide. Therefore, when we remove $\mathcal{L}_{occ}(\mathcal{X}_o)$ from our optimization objective, we observe that the predicted primitives accurately represent geometric parts of the human body but fail to capture the empty space (see hands and legs in first column of Fig. 7). Note that the points on the primitive surface that capture empty space have a small distance from points on the target object and as a result, $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p)$ has small values for these points, which makes it difficult to penalize them for covering unoccupied space.

w/o Normal Consistency Loss: The normal consistency loss enforces that the normals of the predicted primitives will be aligned with the normals of the target object. We notice that when we remove $\mathcal{L}_{norm}(\mathcal{X}_t)$, our model yields non-overlapping primitives that accurately capture the geometry of different human body parts (see Fig. 9). However, at the locations where one primitive meets its adjacent, the predicted primitives have inconsistent normals, hence the connections seem unnatural. This can be better seen in Fig. 8, where we provide close-ups on the locations where two primitives meet and compare with our model. Our model, yields primitives with smooth transitions that look more natural.

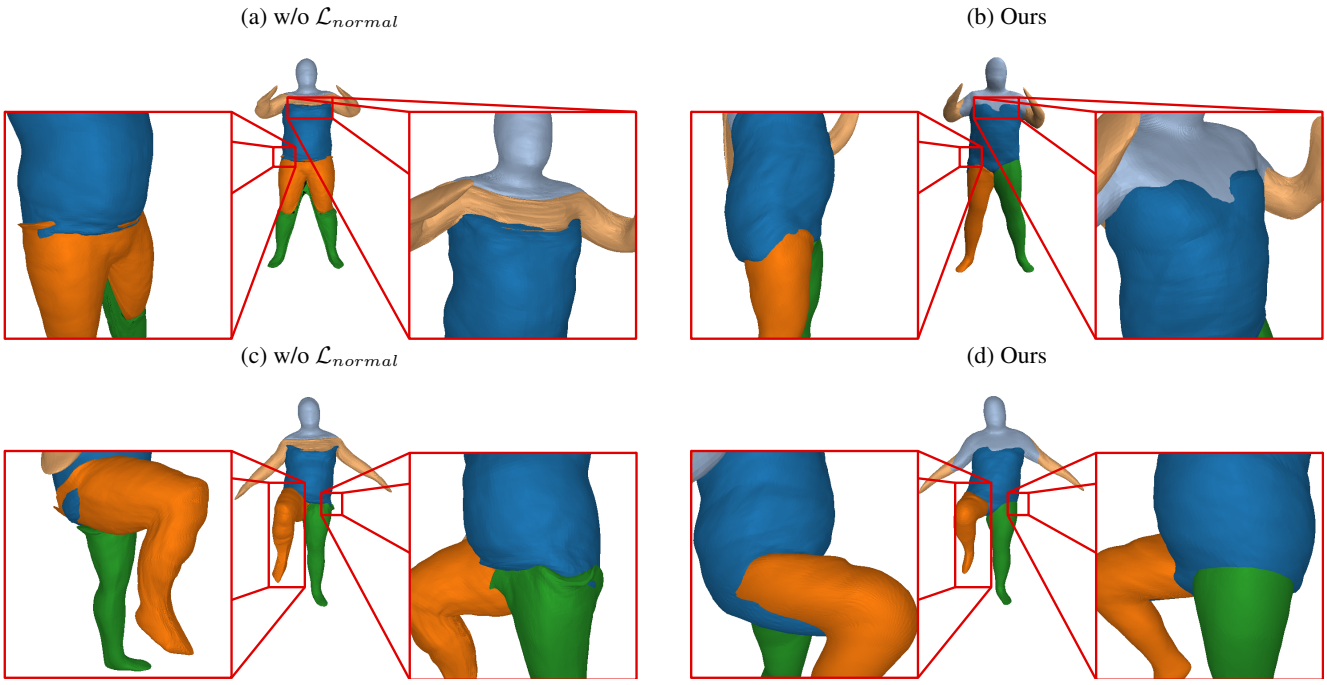


Figure 8: **Impact of $\mathcal{L}_{norm}$.** When removing the $\mathcal{L}_{norm}$ from the optimization objective, the predicted primitives are semantically meaningful but the transitions between one primitive to another are not smooth i.e. do not have consistent normals. This becomes more evident when looking at the locations where primitives meet (marked with a red rectangle). In contrast, our model has smooth transitions and looks more "organic".

w/o Overlapping Loss: The overlapping loss encourages semantically meaningful shape abstractions, where primitives represent distinct geometric parts. To this end, we introduce the $\mathcal{L}_{overlap}(\mathcal{X}_o)$ loss (Eq. 15 in the main submission) to discourage having multiple primitives "containing" the same points from the target object. The reconstructions without the overlapping loss accurately capture the geometry of the human body but fail to yield semantically meaningful shape abstractions, since primitives completely penetrate one another (see Fig. 7). In Fig. 10, we visualize the predicted primitives for various humans and we notice that while they accurately capture the geometry of the human body, they do not correspond to meaningful geometric parts. In particular, there are 2 primitives that describe the right leg (primitive illustrated in blue and orange), 2 for the left leg (primitives illustrated in orange and green) and 3 for the main body (blue, light blue and green). Since, we want our primitives to correspond to semantic parts, this behaviour is not desirable.

w/o Coverage Loss: The coverage loss makes sure that all primitives will "cover" parts of the target object. In practice, it prevents primitives with minimal volume (see Fig. 12, fourth column) that do not contribute to the reconstruction. In Fig. 12, we note that when we remove $\mathcal{L}_{cover}(\mathcal{X}_o)$ from our optimization objective, a primitive degenerates to a "sheet" that contains no points but only has surface area (Fig. 11).

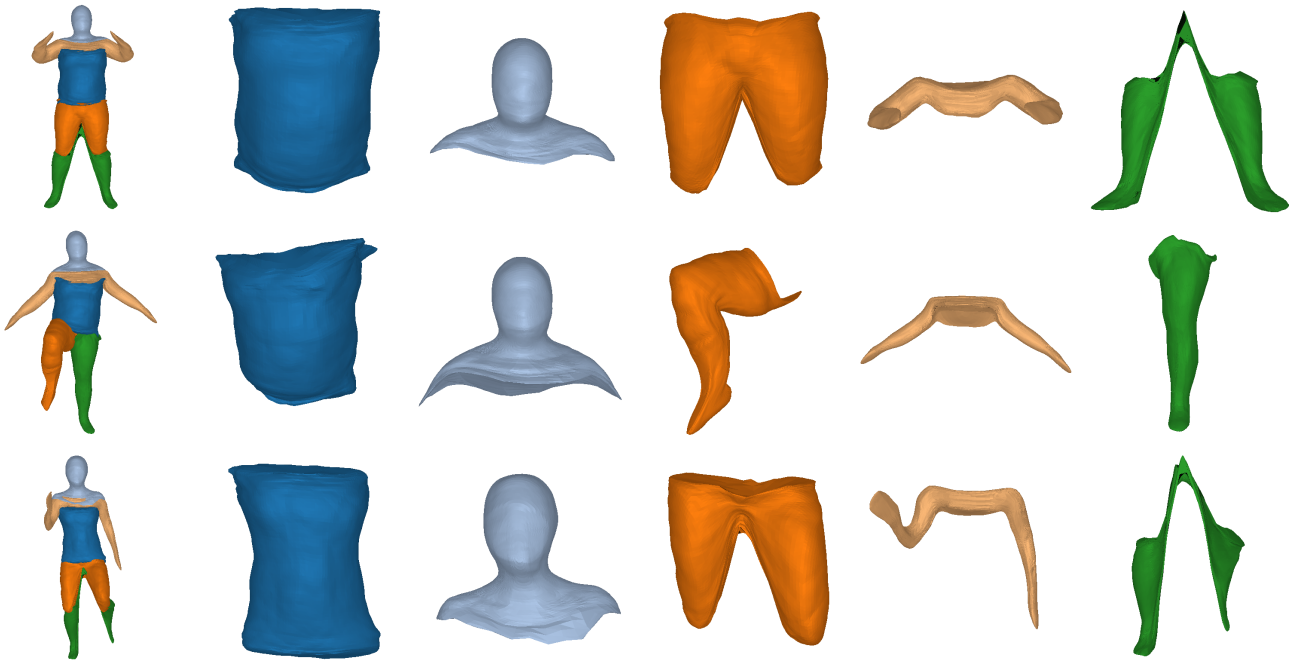


Figure 9: **w/o Normal Consistency Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{normal}$ and we observe that even though the predicted primitives are expressive and have a semantic interpretation, the locations where the primitives meet do not have consistent normals.

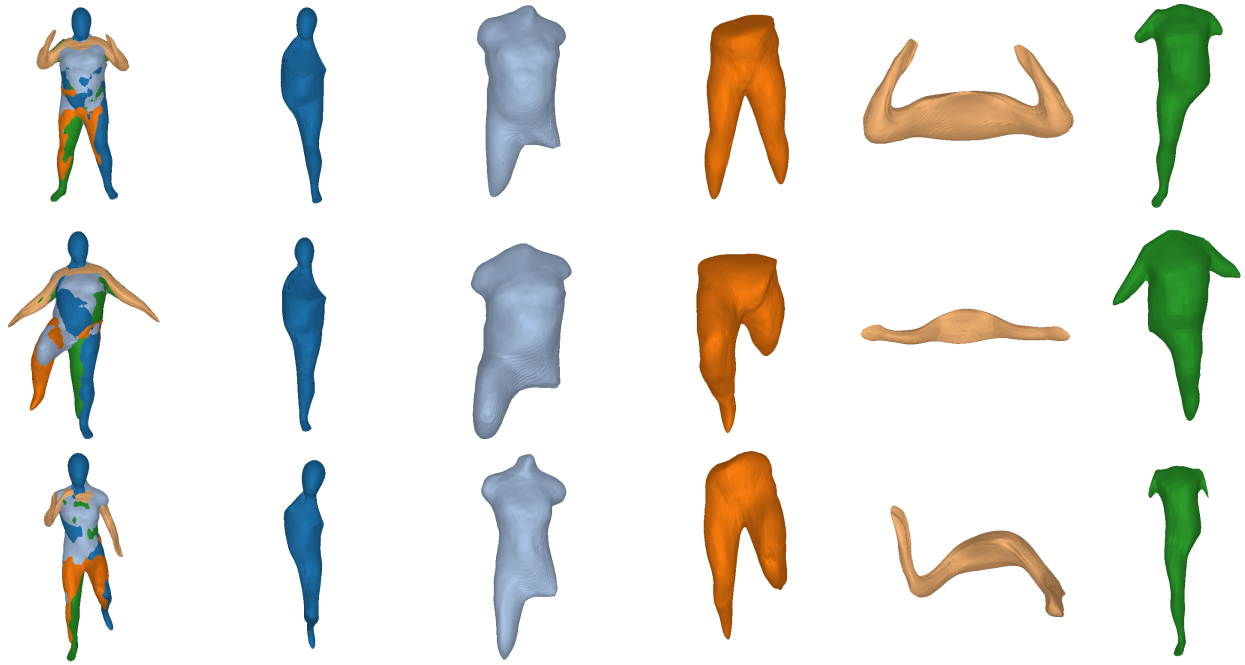


Figure 10: **w/o Overlapping Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{overlap}$ and we observe that even though the predicted primitives are expressive and accurately capture the 3D geometry of the human body, they do not have a semantic interpretation, namely multiple primitives describe the same object part.

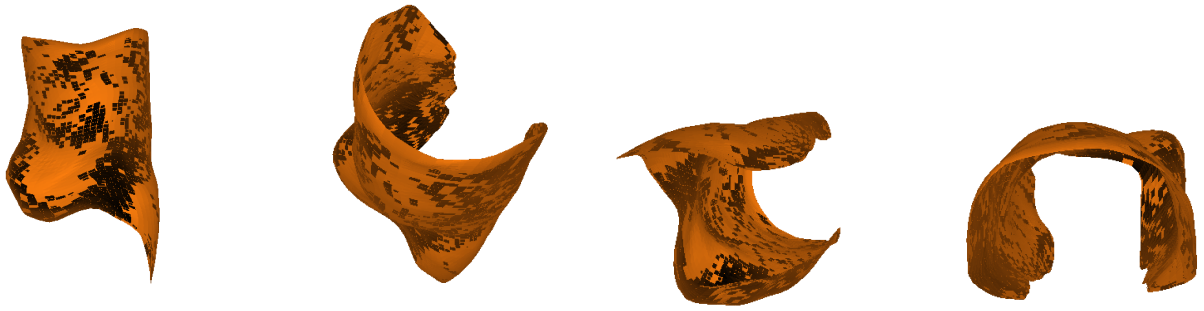


Figure 11: **Impact of $\mathcal{L}_{cover}$.** When removing the $\mathcal{L}_{cover}$ some primitives become very thin (with minimum volume) and do not contribute to the reconstruction loss.

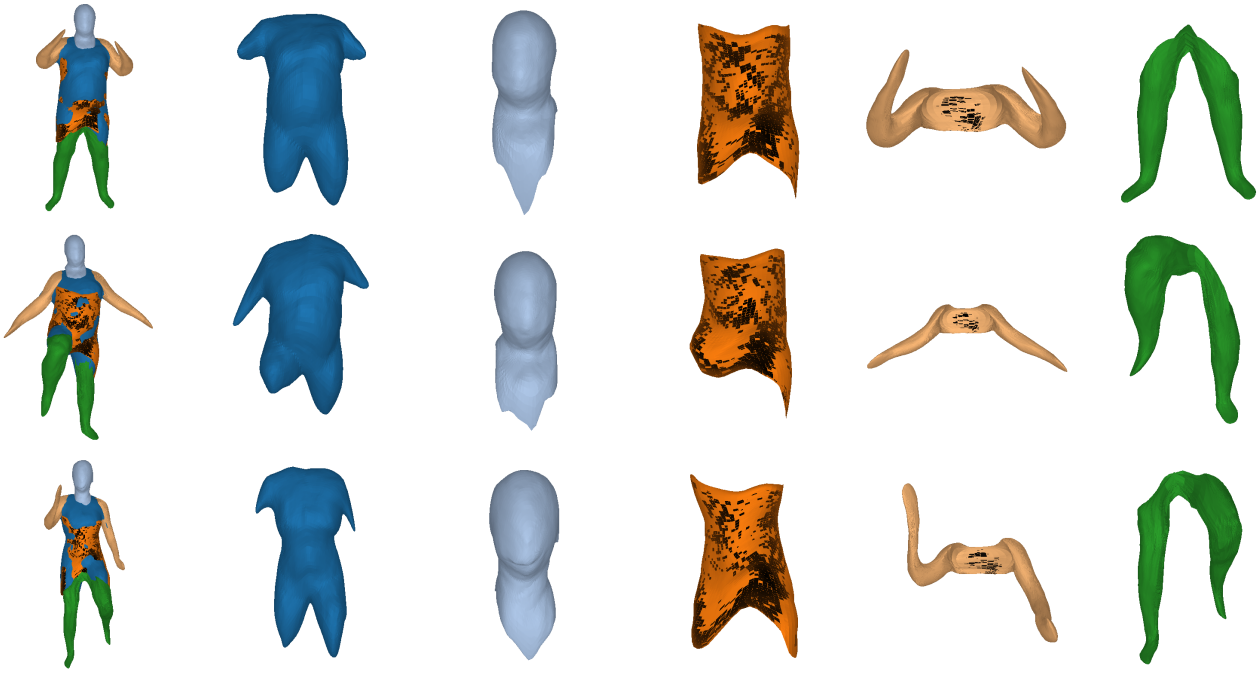


Figure 12: **w/o Coverage Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{cover}$ and observe that although some of them are expressive and efficiently capture the 3D geometry, some primitives have minimum volume hence, do not cover parts of the object and in turn, do not contribute to the reconstruction loss.

2.4. Training time

In this section, we compare the training time of Neural Parts with the baseline methods used in the main paper. In order to compare the methods as fairly as possible, we remove many variables that could add variance in our timing measurements. In particular, we use the same batch size and number of primitives for all methods and measure the time each method requires to perform 100 forward/backward passes. In addition, in order to remove the overhead of the data loading, we load a single batch and keep it in GPU memory. We observe that Neural Parts are slightly slower wrt. time per batch, however, the computational overhead is negligible compared to the data loading overhead and given the improved final performance.

	OccNet	SQs	H-SQs	CvxNet	Ours
Time per batch	56.73 ms (1.16×)	48.64 ms (1×)	63.50 ms (1.32×)	84.36 ms (1.73×)	256.19 ms (5.26×)

2.5. Sensitivity to initialization

In this section, we investigate the sensitivity of our model to initialization, namely we showcase that our model consistently yields almost identical semantic parts for different random initializations. In particular, we train our model three times with three different random seeds on D-FAUST with 6 primitives and visualize the predicted primitives on various humans in Fig. 13. We observe that while some of the predicted parts for Run#1 are slightly different from the parts in Run#2, 3, they are still semantically meaningful.

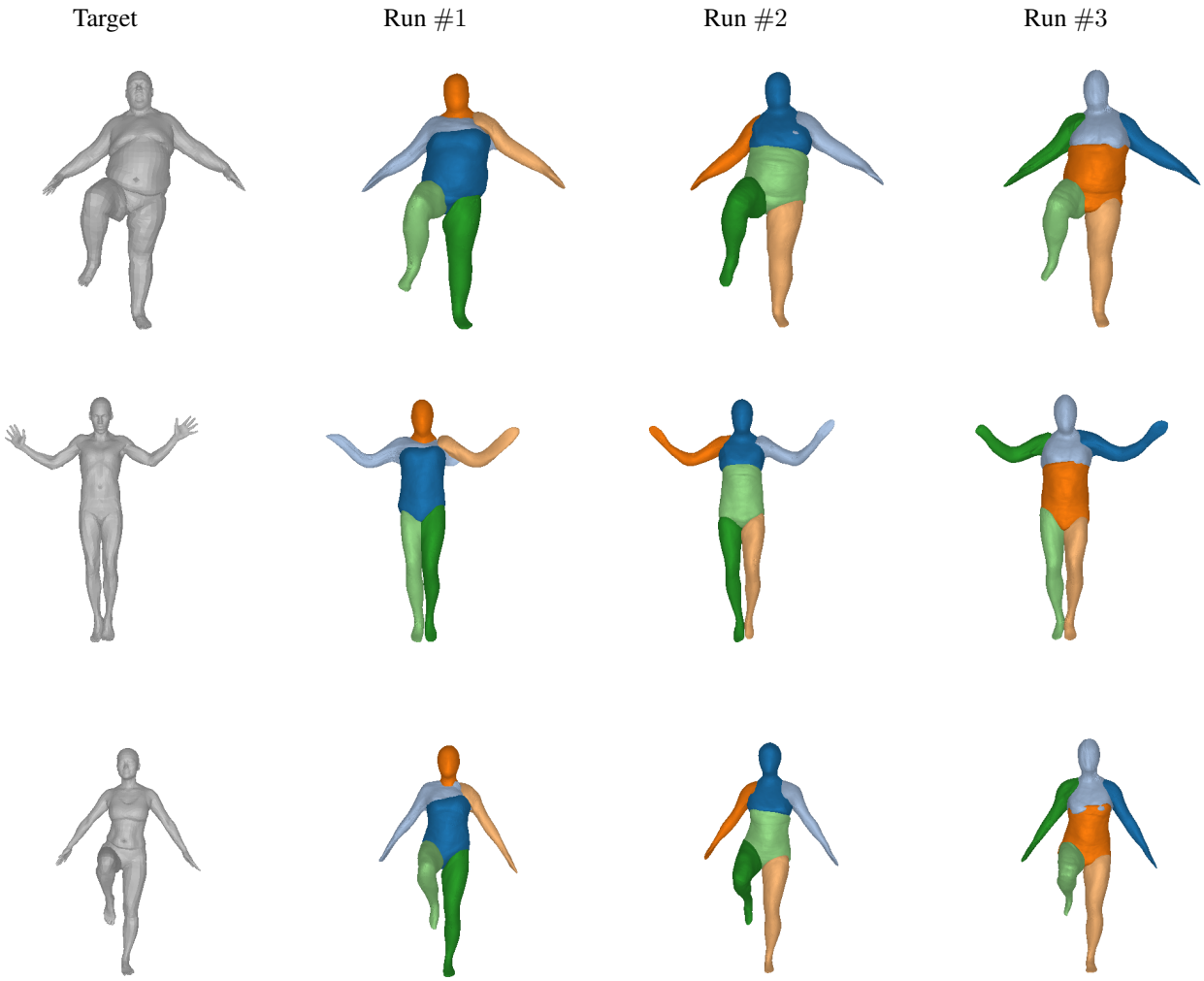


Figure 13: **Sensitivity to Initialization**. The input image is shown on the first column and the rest contain predictions of our method when trained three different times with three different random seeds.

3. Experiment on D-FAUST

In this section, we provide additional information regarding our experiments on D-FAUST dataset [2]. D-FAUST contains $38,640$ meshes of humans performing various tasks such as "chicken wings", "running on spot", "shake arms" etc. We follow [16] and use 70%, 20% and 10% for the train, test and validation splits. Furthermore, we filter out the first 20 frames for each sequence that contain the unnatural "neutral pose" necessary for calibration purposes. Note that in contrast to [16], we do not normalize the meshes to the unit cube in order to retain the variety of the human body i.e. tall and short humans. This makes the single view 3D reconstruction task harder, as all models need to efficiently capture both the pose and the size of the human.

Tab. 4 summarizes the quantitative results from Fig. 4 in our main submission. We notice that for all primitive-based baselines, increasing the number of parts results in improved reconstruction quality in terms of volumetric IoU and Chamfer-L_1 distance. Instead, Neural Parts, decouple the number of primitives from the reconstruction accuracy as the performance of our model is independent of the number of the predicted primitives. This is expected, since our primitives are highly expressive and our model can capture the 3D shape geometry using even a single primitive (see Fig. 14). We argue that this is a desirable property, as Neural Parts enable us to select the number of parts based on the expected number of semantic parts and not the desired reconstruction quality.

		SQs				H-SQs				CvxNet				Ours			
	OccNet	5	10	25	50	4	8	16	32	5	10	25	50	2	5	8	10
IoU	0.691	0.514	0.565	0.616	0.607	0.559	0.595	0.623	0.610	0.582	0.607	0.622	0.621	0.675	0.673	0.676	0.678
Chamfer-L_1	0.102	0.147	0.129	0.114	0.116	0.161	0.148	0.126	0.129	0.183	0.161	0.139	0.121	0.101	0.097	0.090	0.095

Table 4: **Single Image Reconstruction on D-FAUST.** We report the volumetric IoU ($\uparrow$) and the Chamfer-L_1 ($\downarrow$) wrt. the ground-truth mesh for our model compared to primitive-based methods SQs [15] H-SQs [16], CvxNet [6] for various number of primitives and the non primitive-based OccNet [14]. We observe that our model outperforms all primitive-based baselines with only 2 primitives.

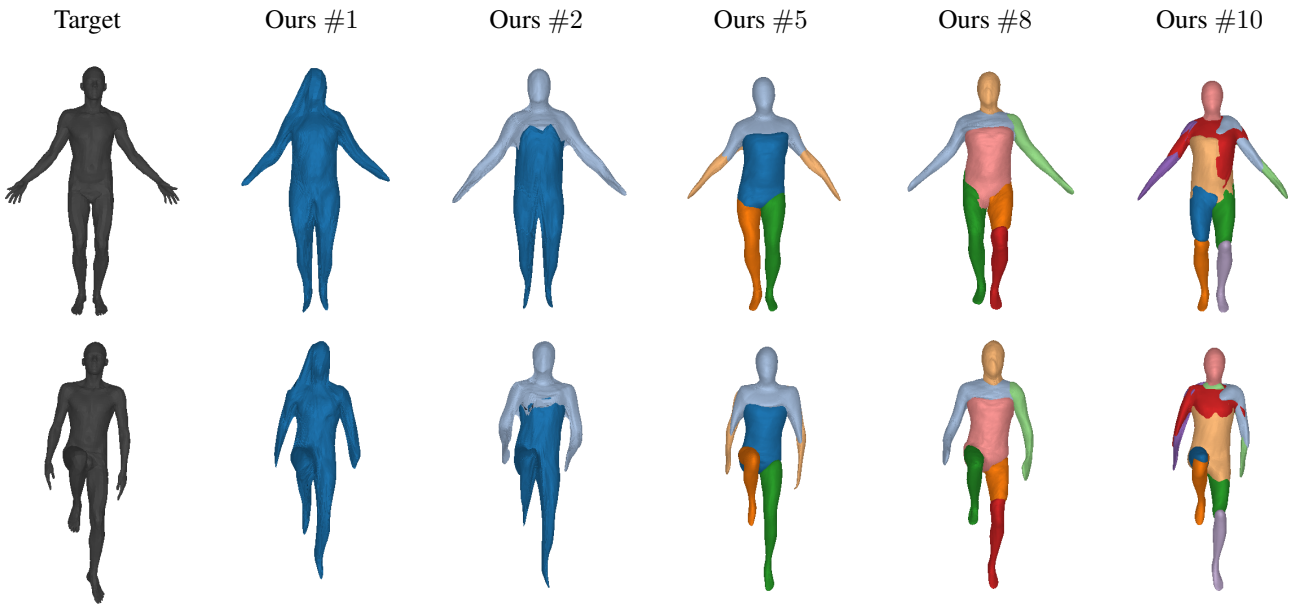


Figure 14: **Single Image 3D Reconstruction on D-FAUST**. The input image is shown on the first column and the rest contain predictions of our method with different number of primitives.

3.1. Additional Experimental Results

In Fig. 15, we provide additional experimental results on the single-view 3D reconstruction task on D-FAUST. Similar, to the results in section 4.2 of the main submission, we compare our model with 5 primitives, with CvxNet and SQs with 50 primitives and H-SQs with 32 primitives. We observe that while CvxNet and H-SQs accurately capture the geometry of the

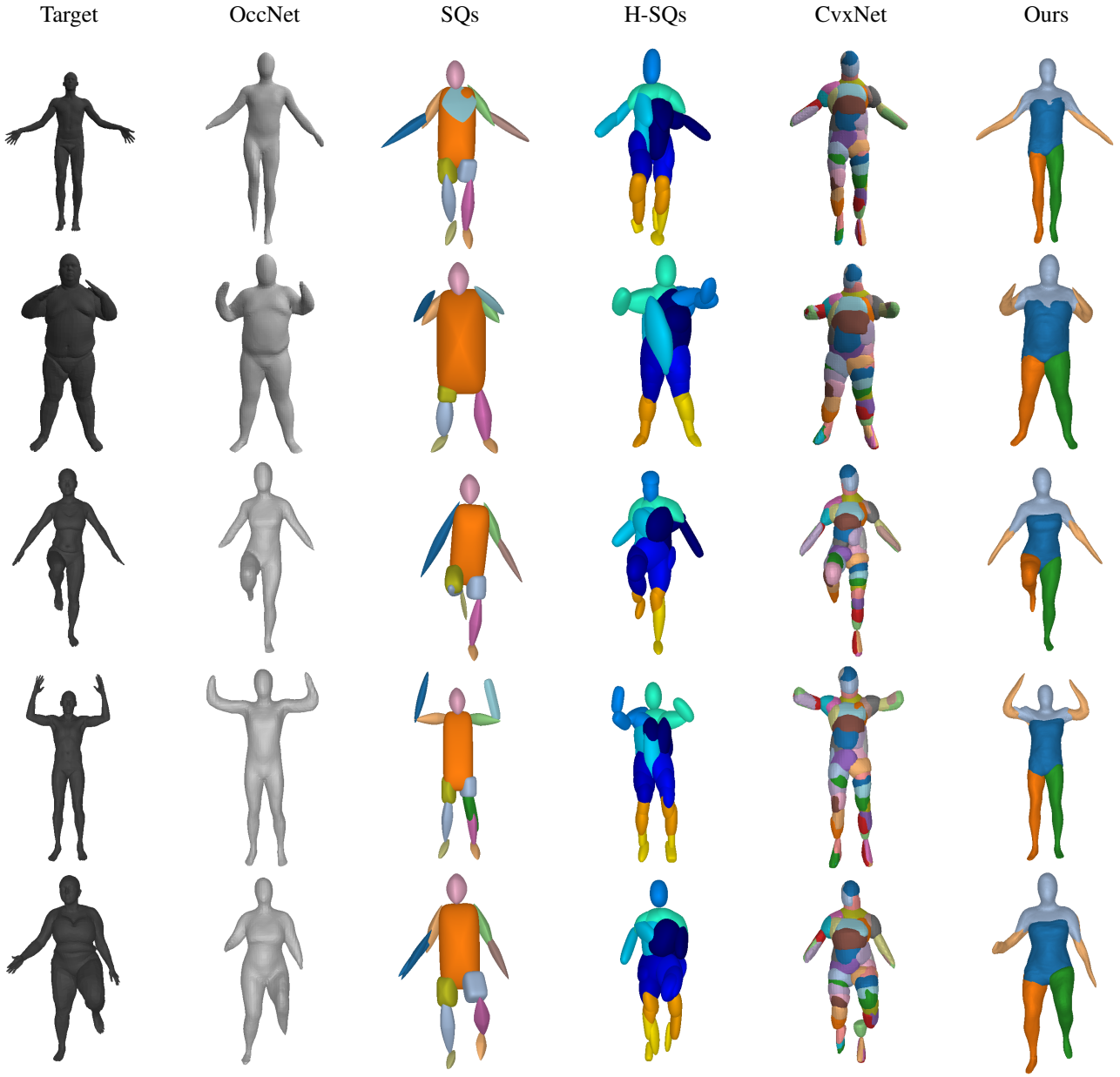


Figure 15: **Single Image 3D Reconstruction on D-FAUST**. The input image is shown on the first column and the rest contain predictions of all methods: OccNet (second), primitive-based predictions with superquadrics (third and fourth) and convexes (fifth) and ours with 5 primitives (last).

human body, their predicted primitives lack any semantic interpretation, as primitives do not correspond to actual geometric parts of the human body. Instead, SQs lead to semantic and parsimonious shape abstractions that are not accurate. On the other hand, Neural Parts achieve both semantic interpretability and high reconstruction quality.

4. Experiment on FreiHAND

In this section, we provide additional information regarding our experiments on FreiHAND dataset [21]. For FreiHAND, we select the first 5000 hand poses and generate meshes using the provided MANO parameters [19]. We render them from a fixed orientation and use 70%, 20%, 10% for the train, test and validation splits. Note that MANO does not generate watertight meshes, thus we need to post-process the meshes in order to create occupancy pairs. This is achieved by adding triangular faces using the following vertex indices $38, 92, 234, 239, 279, 215, 214, 121, 78, 79, 108, 120, 119, 117, 118, 122$.

4.1. Additional Experimental Results

In Fig. 16, we provide additional qualitative results on the single-view 3D reconstruction task on FreiHAND. Similar to the results in section 4.2 of the main submission, we compare our model with 5 primitives to SQs and CvxNet with 5 primitives and H-SQs with 8. Note that for H-SQs we can only use a maximum number of primitives that is a power of 2. H-SQs with 4 primitives performed poorly; thus we increased the number of parts. We observe that Neural Parts yield semantically meaningful parts that accurately capture the hand geometry. Note that accurately modeling objects with non-rigid parts, such as the flexible human fingers, requires primitives that can bend arbitrarily (see thumb in second row Fig. 16). This is only possible with Neural Parts, as we do not impose any kind of constraint on the primitive shape.

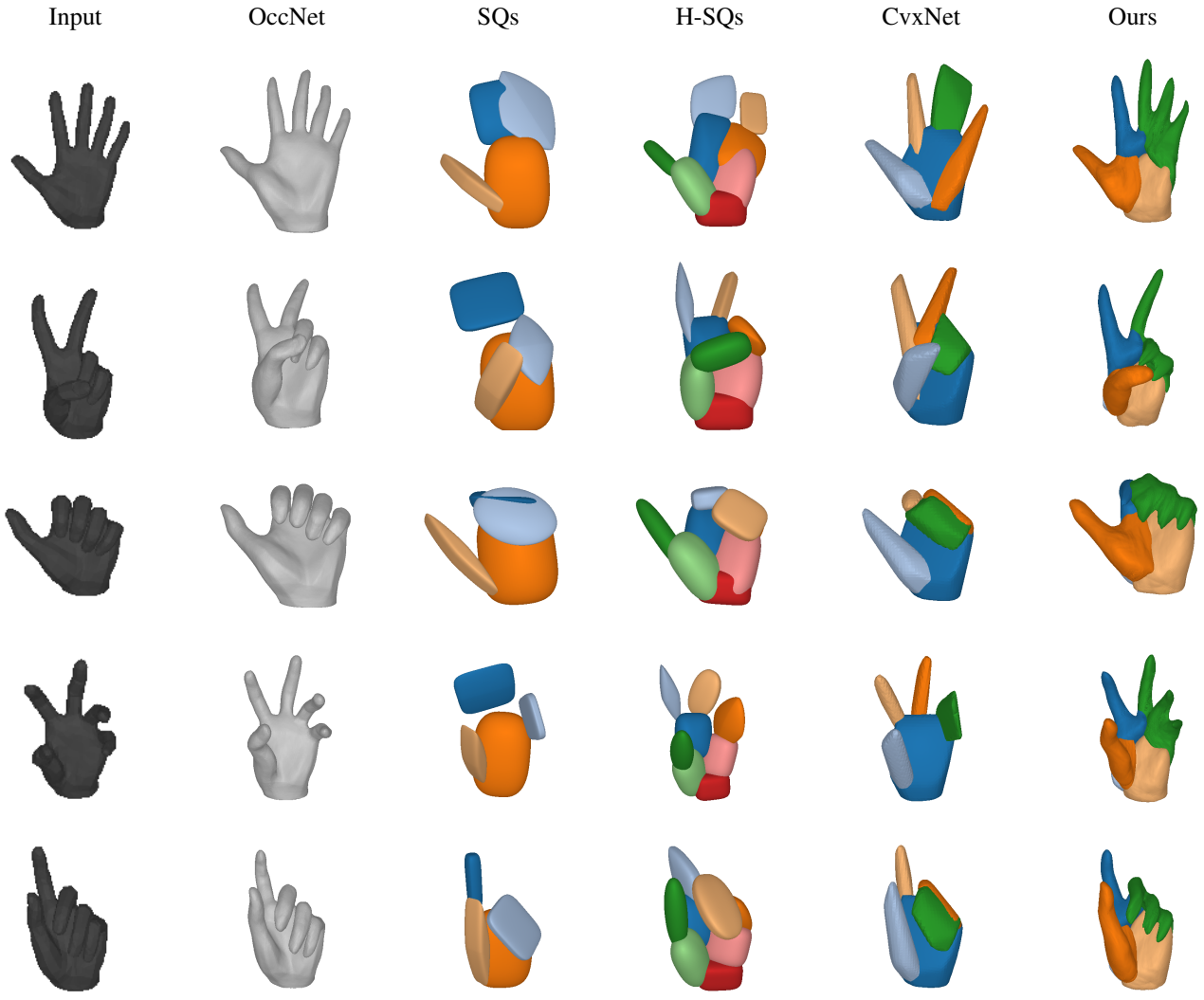


Figure 16: **Single Image 3D Reconstruction on FreiHAND**. We compare our model with OccNet, SQs and CvxNet with 5 primitives and H-SQs with 8 primitives.

5. Experiment on ShapeNet

In this section, we provide additional information regarding our experiments on ShapeNet dataset [3]. In particular, we use the same image renderings and train/test splits of Choy et al. [4]. In order to determine whether points lie inside or outside the target mesh (i.e. for generating the occupancy pairs $\mathcal{X}_o$) we need the meshes to be watertight. For this, we follow [14] and use the code provided by Stutz et al. [20][4] which performs TSDF-fusion on random depth renderings of the object, to create watertight meshes of the object. In Fig. 17, we provide additional qualitative results on various ShapeNet cars and compare our model with 5 primitives to CvxNet with 5 and 25 and with the non primitive-based OccNet.

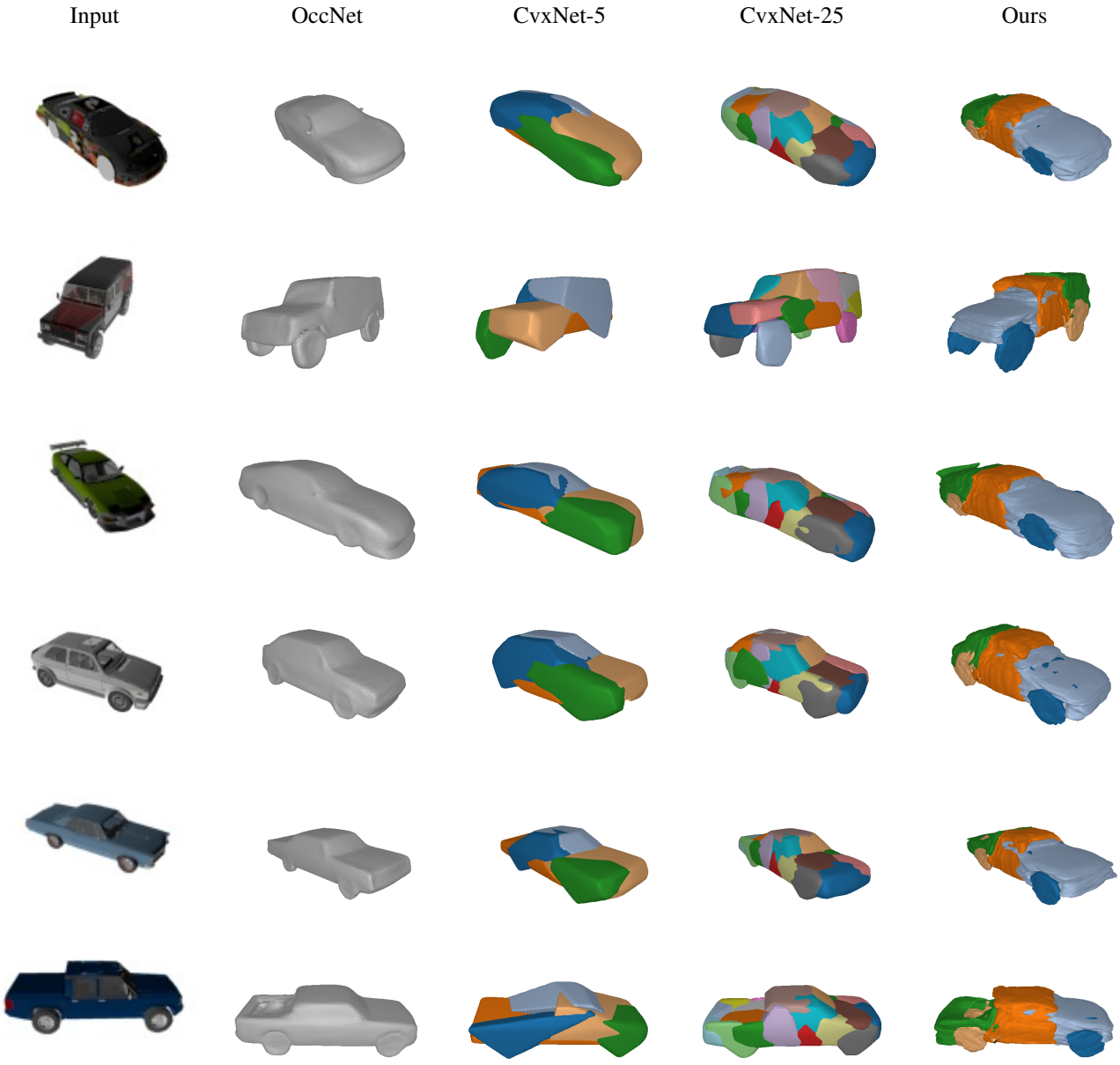


Figure 17: **Single Image 3D Reconstruction on ShapeNet**. We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

[4] https://github.com/davidstutz/mesh-fusion

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.763	0.650	0.666	**0.697**
Chamfer-L_1	0.186	0.218	0.210	**0.185**

Table 5: **Single Image Reconstruction on ShapeNet cars.** Quantitative evaluation of our method against OccNet [14] and CvxNet [7] with 5 and 25 primitives.

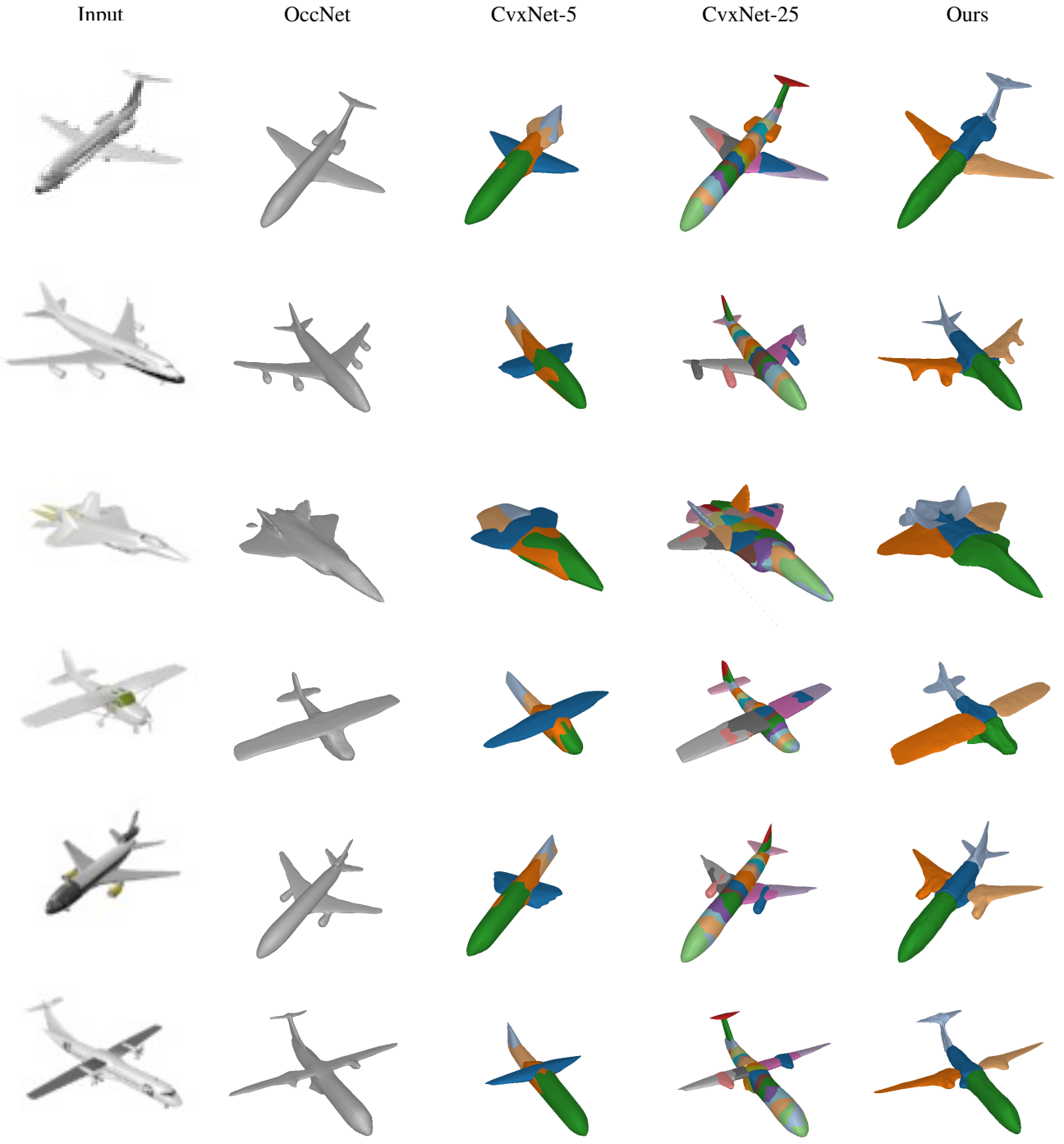


Figure 18: **Single Image 3D Reconstruction on ShapeNet**. We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.451	0.425	0.448	**0.454**
Chamfer-L_1	0.218	0.267	0.245	**0.220**

Table 6: **Single Image Reconstruction on ShapeNet airplanes.** Quantitative evaluation of our method against OccNet [14] and CvxNet [7] with 5 and 25 primitives.

Subsequently, we train Neural Parts with 5 primitives and compare with CvxNet with 5 and 25 primitives on ShapeNet planes (Fig. 18 + Tab. 6), ShapeNet lamps (Fig. 19 + Tab. 7) and ShapeNet chairs (Fig. 20 + Tab. 8). In Fig. 18, we provide a qualitative evaluation of our model with CvxNet and OccNet on various planes. We observe that Neural Parts yield more geometrically accurate and semantically meaningful shape abstractions than CvxNet with multiple primitives, i.e. the same primitive is consistently used for representing the tail and the wings of the airplanes. This is also validated quantitatively in Tab. 6, where we see that our model outperforms CvxNet in terms of IoU and Chamfer-L_1 both with 5 and 25 primitives. We observe that CvxNet with 5 primitives cannot represent fine details and as a result parts of the target object are not captured, e.g. the turbines of airplanes.

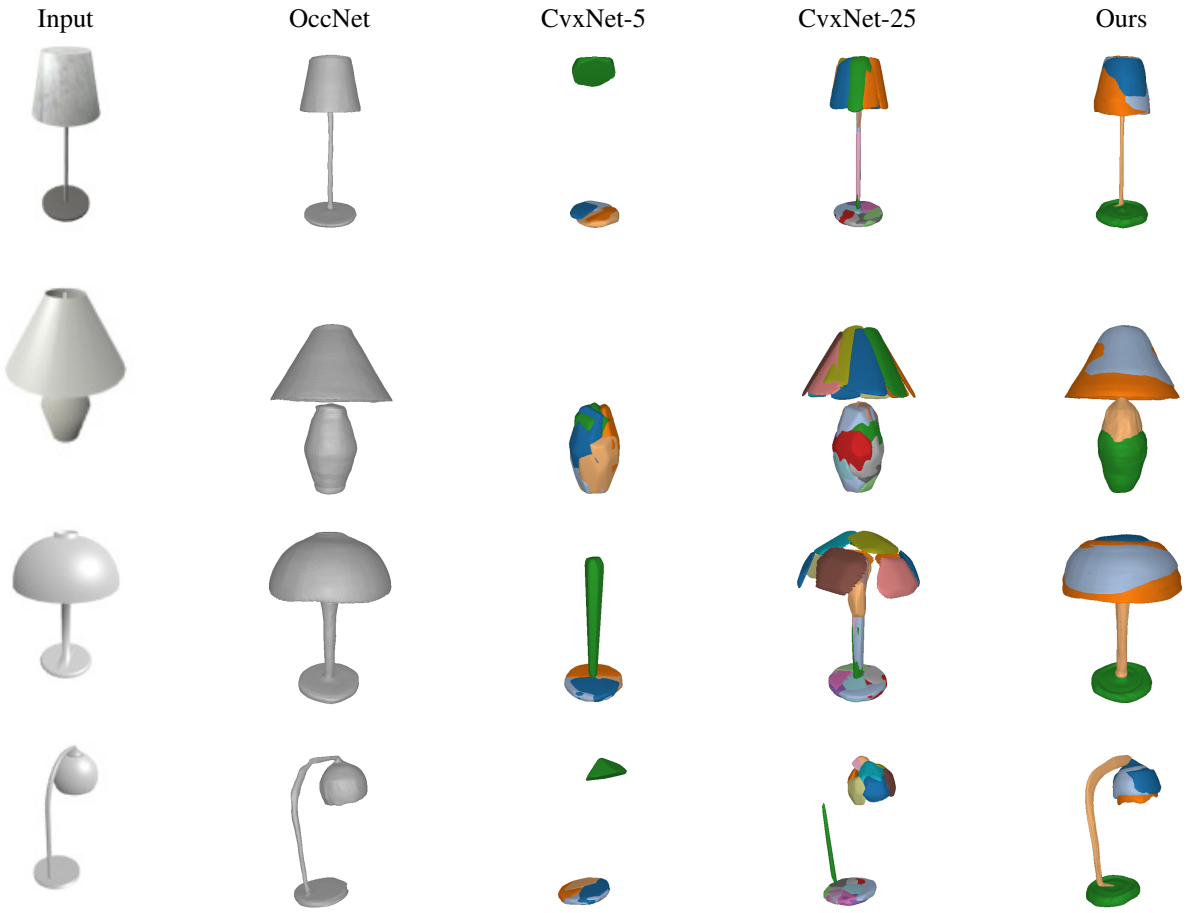


Figure 19: **Single Image 3D Reconstruction on ShapeNet**. We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

This becomes more evident for the case of lamps, where 5 primitives do not have enough representation power for modelling complex shapes such as the lamp shade or the lamp body (see Fig. 19 third column), thus entire object parts are missing. Similarly, also for the case of chairs, CvxNet with 5 primitives fail to accurately capture the object's geometry (see Fig. 20 third column). For example, 5 primitives are not enough for representing the legs of the chair. Instead our model

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.339	0.246	0.278	**0.318**
Chamfer-L_1	0.514	0.613	0.537	**0.347**

Table 7: **Single Image Reconstruction on ShapeNet lamps.** Quantitative evaluation of our method against OccNet [14] and CvxNet [7] with 5 and 25 primitives.

with the same number of parts consistently captures the 3D geometry. On the contrary, CvxNet with 25 primitives yield more geometrically accurate reconstructions, however, the reconstructed primitives are not as semantically meaningful.

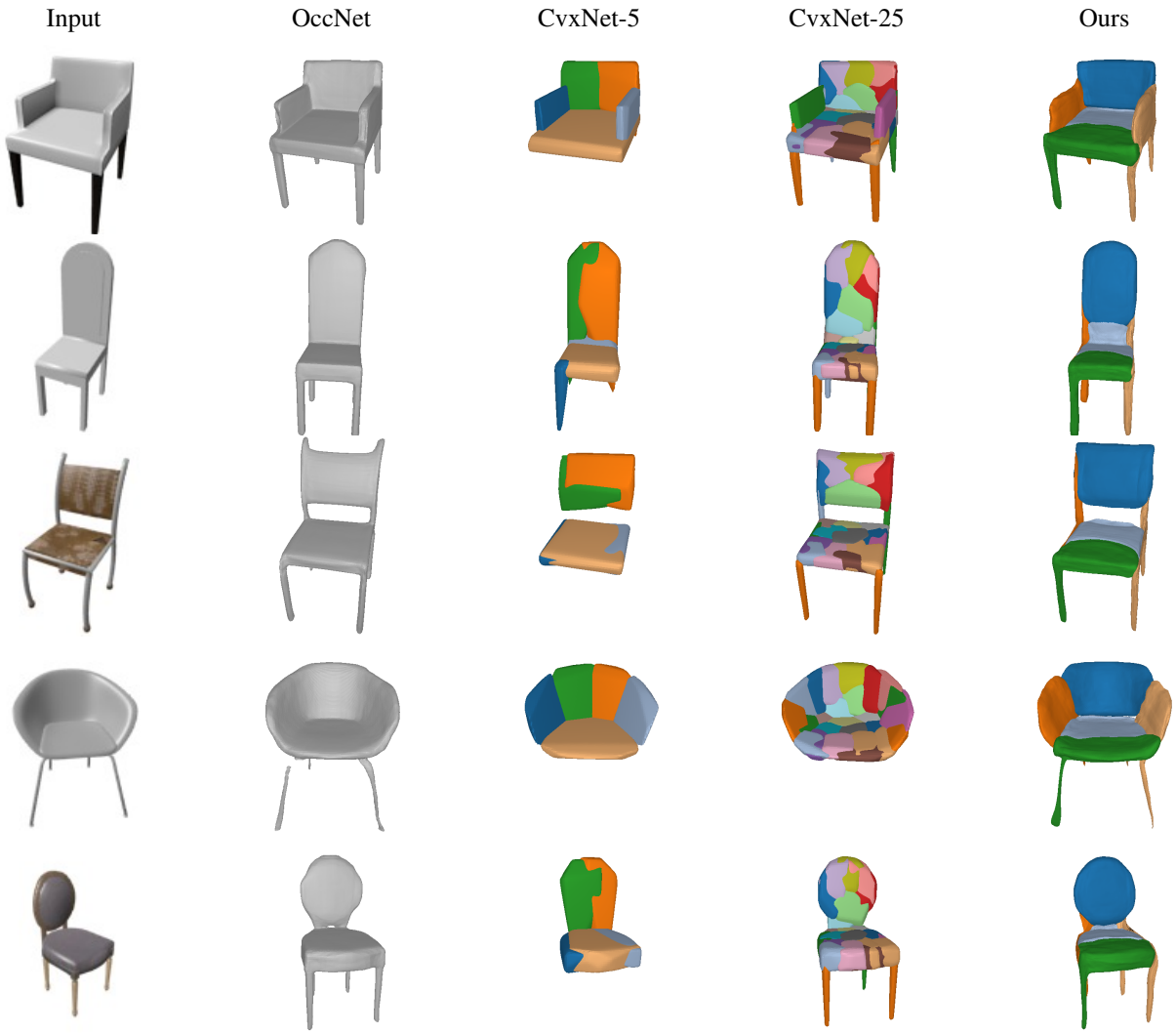


Figure 20: **Single Image 3D Reconstruction on ShapeNet**. We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

6. Semantically Consistent Abstractions

In this section, we provide additional information regarding our Semantic Consistency experiment from Section 4.3 in our main submission. Furthermore, since D-FAUST contains 4D scans of humans in motion, we also evaluate the temporal consistency of the predicted primitives on various actions.

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.432	0.364	0.392	**0.412**
Chamfer-L_1	0.312	0.587	0.557	**0.337**

Table 8: **Single Image Reconstruction on ShapeNet chairs.** Quantitative evaluation of our method against OccNet [14] and CvxNet [7] with 5 and 25 primitives.

Figure 21: **Representation Consistency.** Our predicted primitives are consistently used for representing the same human part for different humans.

6.1. Temporal Consistency

In this section, we provide a qualitative analysis on the temporal consistency of the predicted primitives. Results are summarized in Fig. 22. We observe that Neural Parts consistently use the same primitive for representing the same object

part regardless of the breadth of the part's motion. Notably, this temporal consistency is an emergent property of our method and not one that is enforced with any kind of loss. Additional visualizations are provided in the supplementary video.

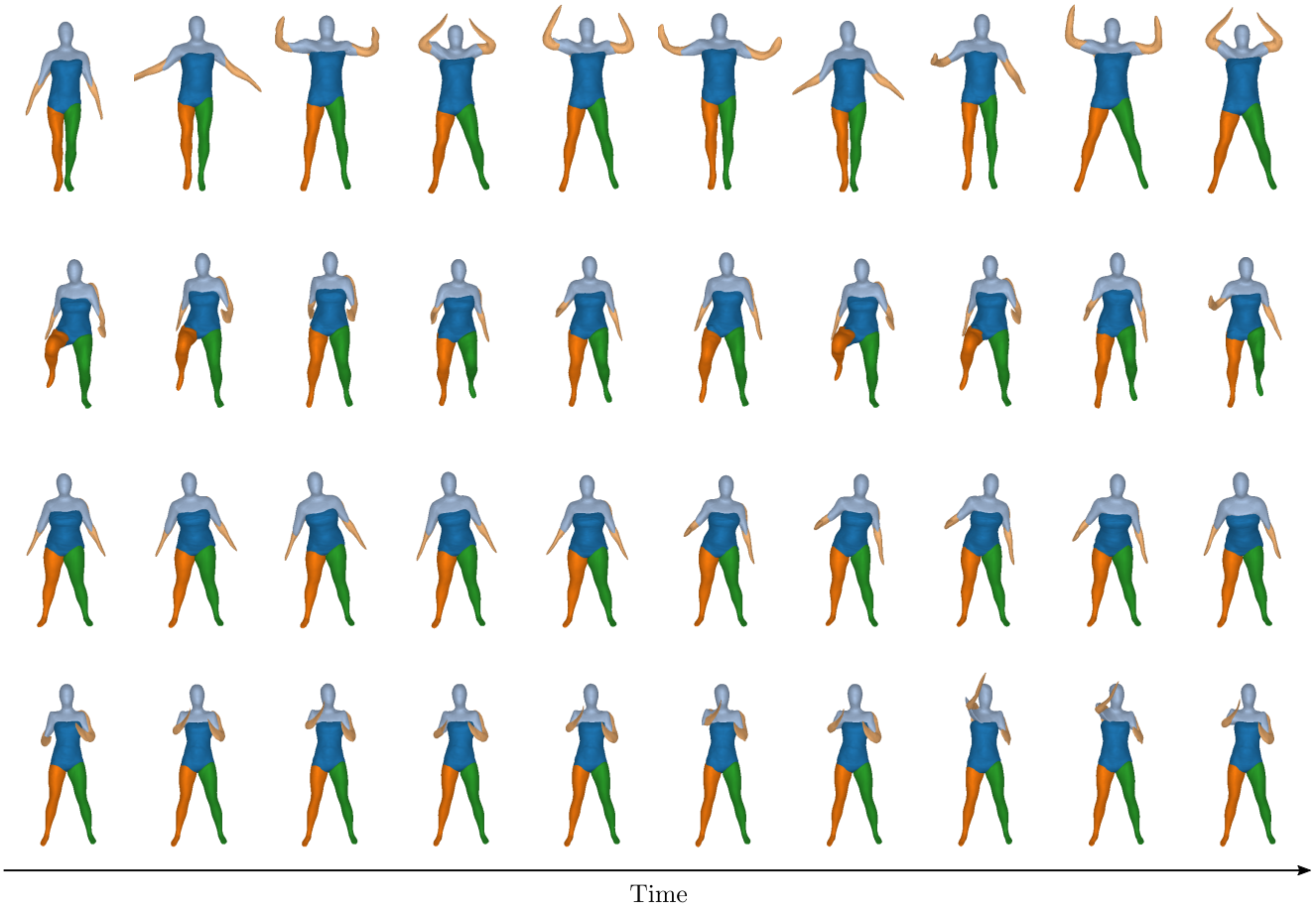


Figure 22: **Temporal Consistency of Predicted Primitives.** We note that Neural Parts yield primitives that preserve their semantic identity while different humans perform various actions.

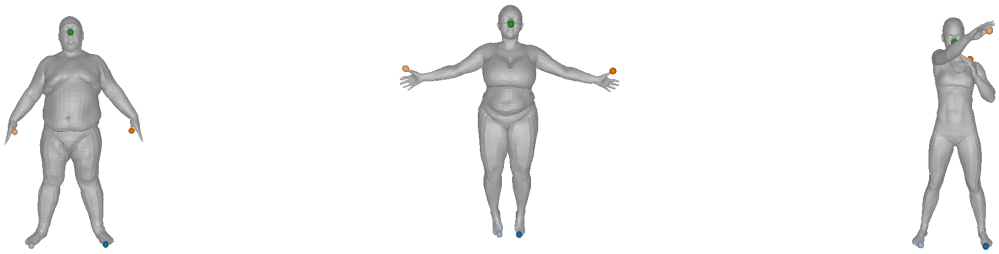


Figure 23: **Semantic vertices provided by SMPL-X.** We highlight the 5 vertex indices that we use to identify left thumb (L-thumb), right thumb (R-thumb), left toe (L-toe), right toe (R-toe) and nose.

6.2. Semantic Consistency

In this experiment, we provide additional information regarding the Semantic Consistency experiment in our main submission (see Fig. 10). In particular, we evaluate whether a specific human part is represented consistently by the same primitive (see Fig. 21). To measure this quantitatively, we select 5 representative vertices on each target mesh (the vertex

indices are provided by SMPL-X [18]) and measure the classification accuracy of those points when using the label of the closest primitive. A visualization of the 5 vertices on the human body is given in Fig. 23. Note that the vertex indices are consistent across all subjects because D-FAUST meshes are generated using SMPL [13].

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. *arXiv.org*, 1605.08695, 2016. 4

[2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: registering human bodies in motion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 13

[3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 1, 16

[4] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 16

[5] A. H. J. Christensen. A note on geodesic polyhedra: Triangulation and contouring of spheres. *Comput. Graph.*, 3(4):163–165, 1978. 3

[6] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *arXiv.org*, 2019. 13

[7] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 17, 18, 19, 20

[8] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. 1, 6

[10] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 3

[13] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. on Graphics*, 2015. 22

[14] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4, 13, 16, 17, 18, 19, 20

[15] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4, 13

[16] Despoina Paschalidou, Luc van Gool, and Andreas Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 13

[17] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv.org*, 1606.02147, 2016. 4

[18] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 22

[19] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: modeling and capturing hands and bodies together. *ACM Trans. on Graphics*, 36(6):245:1–245:17, 2017. 15

[20] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 16

[21] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 15