GLASS: Geometric Latent Augmentation for Shape Spaces

Sanjeev Muralikrishnan[1], Siddhartha Chaudhuri[2,3], Noam Aigerman[2], Vladimir Kim[2], Matthew Fisher[2], and Niloy Mitra[1,2]

[1]University College London
[2]Adobe Research
[3]IIT Bombay

Figure 1: Starting from just 10 shapes (larger, numbered), our method iteratively augments the collection by alternating between training a VAE, and exploring random perturbations in its low-dimensional latent space guided by a purely geometric deformation energy. Here we show the 1000 most diverse shapes from the first 2.5K discovered by our method, positioned according to their latent embedding (projected to 2D via t-SNE). Shapes are colored according to the initial landmark they trace back to, with shapes added in later iterations lighter (greyer) in color. The augmentation effectively fills in the space between the sparse initial landmarks, and even extrapolates beyond them. It manages to also interpolate global rotations for samples near the back-facing 3rd exemplar, and yields crossed-legged models even though there is no such initial landmark.

Abstract

We investigate the problem of training generative models on a very sparse collection of 3D models. We use geometrically motivated energies to augment and thus boost a sparse collection of example (training) models. We analyze the Hessian of the as-rigid-as-possible (ARAP) energy to sample from and project to the underlying (local) shape space, and use the augmented dataset to train a variational autoencoder (VAE). We iterate the process of building latent spaces of VAE and augmenting the associated dataset, to progressively reveal a richer and more expressive generative space for creating geometrically and semantically valid samples. Our framework allows us to train generative 3D models even with a small set of good quality 3D models, which are typically hard to curate. We extensively evaluate our method against a set of strong baselines, provide ablation studies and demonstrate application towards establishing shape correspondences.

We present multiple examples of interesting and meaningful shape variations even when starting from as few as 3-10 training shapes.

1. Introduction

This paper is concerned with generating plausible deformations of a 3D model from a very sparse set of examples. Figure 1 shows an input of 10 example deformations of a 3D mesh of a human in different poses, and the additional deformations generated by our method.

3D deformations have a strong *semantic* element to them – a human's limbs should only bend at the joints, and then not beyond the 180° that the joints permit. Arguably, this

can only be deduced via learning by example from a dataset.

Unfortunately, as opposed to 2D images, the 3D domain poses several challenges for the data-driven framework: meshes come in a non-uniform representation, each mesh with its own unique triangulation, making it hard to devise a data-driven technique that generalizes to different models in different triangulations; furthermore, data acquisition is much more tedious, making datasets scarcer.

Hence, given the scarcity of data, we set to tackle the challenge of generating additional meaningful deformations from a given (very) sparse set of deformations of a single model. The output of our method can then be used to create larger datasets that in turn can be leveraged by other techniques that cannot operate on scarce datasets.

Our core idea is to use supervised learning of a generative space from a training dataset, but augment that dataset in an unsupervised, geometry-aware way. Namely, our main contribution stems from observing that while meaningful deformations are in essence semantic, they also possess a very strong pure-geometric element, e.g., they are smooth (i.e., preserve local details) and do not distort the model too much (i.e., local lengths of elements are relatively preserved). Leveraging this observation enables us to reduce the dimensionality of the problem, in turn enabling us to devise a generative method which requires a very small amount of examples as input.

Specifically, we train a Variational Autoencoder (VAE) on the given dataset. During its training, we consider random latent codes and the decoded deformation they represent. We propose an unsupervised geometry-aware method, using the eigenmodes of the deformation energy's Hessian, to *perturb and project* existing latent codes in a way that ensures they yield smooth, low-distortion deformations, which we add as data-augmentation to the input datasets. We then train the VAE on the augmented dataset and repeat the process iteratively until the latent space has been sampled densely enough. We call our method GLASS.

We evaluate GLASS on a selection of established datasets (e.g., FAUST, centaur, horse) and compare performance on different baseline alternatives using a combination of evaluation measures. The experiments show the effectiveness of GLASS to recover meaningful deformation from a mere handful of exemplars. We also evaluate the the proposed method in the context of shape correspondence, demonstrating the utility of the sampling process to assist the 3D-CODED [15] algorithm.

2. Related Work

Geometric shape deformation. *Parametric* deformation methods express 2D or 3D shapes as a known function of a set of common parameters, and model deformations as variations of these parameters. Such methods include cages [20], blendshapes [25], skinned skeletons [19] and Laplacian eigenfunctions [28]. In contrast, *variational* methods model deformations as minimizers of an energy functional – e.g. Dirichlet [16], isometric [21], conformal [24], Laplacian [5], As-Rigid-As-Possible (ARAP) [32], or As-Consistent-As-Possible (ACAP) [12] – subject to user constraints. In our work we focus on minimizing the ARAP energy, although our method supports any twice-differentiable energy function. There are strong connections between the parametric and variational approaches, for instance biharmonic skinning weights [18] (parametric) are equivalent to minimizing the Laplacian energy (variational). Please see surveys [23, 38] for a complete discussion. We are also inspired by work on modal analysis [17], which linearize the deformation space of a shape in terms of the least-significant eigenvectors of the Hessian of some energy functional. In the current paper, we effectively perform *learned non-linear modal analysis*: starting with a variational formulation – the implicitly-defined manifold of low-energy perturbations of a few landmark shapes – we *learn* the corresponding parametric representation as the latent space of an autoencoder by iteratively exploring locally linear perturbations.

Our work on data augmentation from a sparse set of landmark shapes is related to *interpolation/morphing* between, and *extrapolation* from, sets of shapes. As in our scenario, the set typically comprises articulations of a common template. See e.g. [35] for a survey of classical (non-learning-based) methods for shape interpolation. Plausible extrapolation is less well-defined, and less studied, in the classical literature. Kilian et al. [21] extend geodesics of an isometric energy in the deformation space, though it is restricted to exploring (and extrapolating) paths between shapes rather than the full deformation space.

Learned deformation models. Various types of generative models based on graphical models, GANs, VAEs etc have been developed to probabilistically synthesize shape variations. A full treatment is beyond the scope of this paper, please see surveys such as [9]. Here, we focus on models which capture the space of smooth deformations of a given shape. The best-studied domain is that of virtual humans, beginning with seminal works capturing face [3], bodyshape [1] and pose [2] variations in a data-driven fashion from scanned exemplars. These works, like several subsequent ones, rely on variations of principal component analysis (PCA) to parameterize the deformation space. Yumer et al. [39] learn a common set of deformation handles for a dataset. More recent work uses deep neural networks to learn shape deformation models from training sets [27, 33, 13, 11, 36], and use them for applications such as non-rigid correspondences [15]. Similar to one aspect of our work, Tan et al. [34] regularize a VAE with an energy-based loss. However, the primary role of the energy in our method is to guide exploration for data augmentation.

Crucially, all the above methods rely on extensive training data. In contrast, we specifically aim to learn meaningful data-driven deformation models under *extreme sparsity constraints*, from just a handful of landmarks indicating modes of the distribution. While this is broadly related to few-shot learning scenarios, only a few other papers consider these requirements in the context of geometric shape synthesis, or without any auxiliary data from other domains. LIMP [10] is an important recent work that tries to regularize the latent space of a 3D shape VAE by requiring points sampled on the line segment between two latent codes to minimize geometric distortion relative to the endpoints. Unlike our method, LIMP does not explore the full volume of the hull bounding the training landmarks, or extrapolate beyond it – regularization is limited to the web of pairwise paths. We modified LIMP to work with ARAP energy, and demonstrate that our method significantly outperforms their approach on a variety of metrics.

Unsupervised data augmentation. Our work is part of a wide class of methods for synthetically increasing the size of training datasets for data-hungry machine learning, without additional supervision. For broad coverage, we refer the reader to surveys on images [31], time series [37], and NLP [8]. A particularly relevant recent technique is Deep Markov Chain Monte Carlo [29], which samples perturbations of training data using MCMC on an energy functional, trains an autoencoder on these samples, and uses the resulting latent space for lower-dimensional (and hence faster) MCMC. We observed that on very sparse and high-dimensional datasets (only a few landmark 3D shapes), the initial samples of Deep MCMC do not capture meaningful variations, and hence it does not adequately augment the dataset. Also related are methods that augment classification datasets with adversarial perturbations along the gradients of loss functions [14, 30]. In contrast, we seek to *preserve* an energy-based loss, and hence *eliminate* the gradient and other high-change directions from consideration.

3. Method

We now describe our method for generating a dense space of deformations from a given sparse set of example deformations. We first describe the general problem setup in Section 3.1, and then move on to the crux of our method, deformation-aware data-augmentation in Section 3.2.

3.1. Problem setup

We assume we are given a mesh with N vertices $V \in \mathbb{R}^{N\times 3}$, and triangles T. A deformation of the mesh is simply an assignment of a new position to each vertex, denoted $W \in \mathbb{R}^{N\times 3}$. We are also given a sparse set of deformation "examples", $W^1, \dots W^k$. Lastly, we assume to possess a deformation energy $f(W)$, which quantifies how severe is a deformation by measuring distortion. We use the As-Rigid-As-Possible (ARAP) energy [32] to measure how much does any deformation strays from isometry, i.e., how much does geodesic lengths changes with respect to the rest pose V. We note, however, that our method can be used with other (piecewise) differentiable energies.

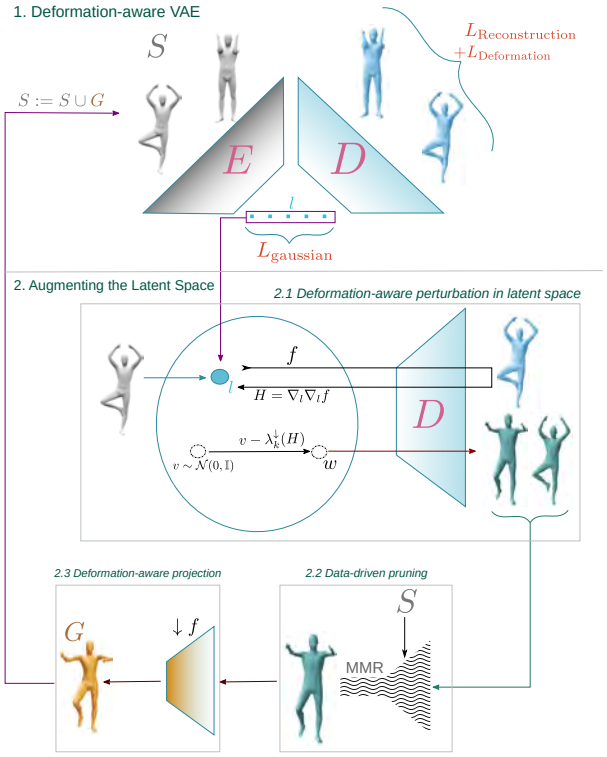

Figure 2: We present GLASS to iteratively build a deformation-aware VAE latent space and analyzing it to generate new training samples to augment the original training set. These enables generation of diverse yet plausible shape variations starting from very few input examples.

We seek a generative method to produce meaningful deformations, which is controllable by the given sparse set. To that end, we devise a subspace-sampling strategy that adheres to two properties: first, it should be data-driven, and contain deformations guided by the given sparse set; and second, it should be geometrically-meaningful, i.e., the deformations should have low energy, with respect to the given deformation energy $f(W)$.

Our approach consists of a data-augmentation method to train a variational autoencoder (VAE) [22]. During training, it explores the sample space, guided by the deformation energy $f(W)$, to discover additional meaningful deformations. These are then used as additional sample points to form an augmented dataset that is used to retrain the VAE, and the process is iterated. We now present the steps.

3.2. Deformation-aware VAE

To denote the two parts of the autoencoder, let $E : \mathbb{R}^{N\times 3} \to \mathbb{R}^K$ be the encoder, mapping a deformation W into vectors of mean and variance, in standard VAE fashion, into a distribution $E(W) = \mu, \Sigma$. These vectors define the mean and variance of a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ from which the latent code of dimension K is sampled, $z \sim \mathcal{N}(\mu, \Sigma)$. Similarly, let $D : \mathbb{R}^K \to \mathbb{R}^{N\times 3}$ be the decoder mapping the latent code to a deformation, $D(z) = W$. We shall slightly abuse notation and use $D(E(W))$ to denote the full autoencoding process of W, including the step of sampling from the Gaussian. We define three losses to be used in training.

(i) Reconstruction Loss: We require that the VAE indeed reconstructs, in expectation, a given input deformation,

$$L_{\text{Reconstruction}} := \|D(E(W)) - W\|^2. \tag{1}$$

(ii) Gaussian Regularizer Loss: Instead of applying the KL divergence loss on the distribution parameters, we constrain the sample mean and covariance of the mini-batch to that of a Unit Gaussian. We found that for small sample size this batch-based loss is better for convergence compared to KL-Divergence. Thus, our Gaussian Regularizer loss is

$$L_{\text{Gaussian}} := \frac{1}{b}\sum_{i=1}^{b}\left(\|\mu_i\|^2 + \|\Sigma_i - \mathbb{I}\|^2\right), \tag{2}$$

where b is the mini-batch size, μ_i, σ_i are the predicted mean and covariance for the i-th sample in the mini-batch, and $\mathbb{I}$ is the identity matrix.

(iii) Deformation Energy: Lastly, we require the resulting deformation to have low deformation energy:

$$L_{\text{Deformation}} := f(W). \tag{3}$$

In summary, our network training loss is,

$$L := L_{\text{Reconstruction}} + L_{\text{Gaussian}} + \alpha L_{\text{Deformation}} \tag{4}$$

where α is a scalar weighting factor applied to the energy function. In our experiments, we set $\alpha = 1e-6$.

3.3. Augmenting the Latent Space

We now move on to describe the crux of our technique – adding additional deformation examples W^j to the latent space to reinforce training. Simply optimizing the above loss for the given sparse set of deformations will lead to slow and sub-optimal training for large meshes. Instead, we continuously introduce new low-energy samples into the training set, by which we make the *data term aware of the deformation energy*. We achieve this through three steps: first, deformation-aware *perturbation* of the latent code in directions that locally *least* modify the deformation energy; second, data-driven *pruning* perturbed codes that do not introduce variance to the current dataset; and third, deformation-aware *projection* of the new codes to further lower their deformation energy. Figure 3 illustrates how the latent space is populated with new deformations over iterations, where colors indicate the base shapes.

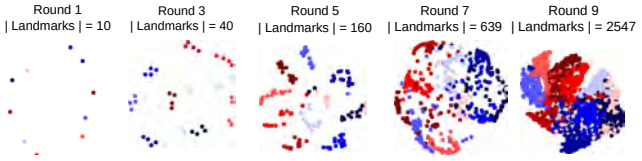

Figure 3: t-SNE embedding of generated samples shows progressive augmentation of the shape space. Sample color indicates originating (parent) shape. See also Fig. 1.

procedure GLASS(W,E,D,f,R)
▷ E = Encoder, D= Decoder
▷ W = Deformation, f = Energy
▷ R = Previously generated shape set
  $l = E(W)$ ▷ latent code
  $g_u = \nabla_l f(D(l))/\|\nabla_l f(D(l))\|$ ▷ unit gradient
  $H = \nabla_l \nabla_l f(D(l))$ ▷ Hessian
  $U, S, V = svd(H)$
  $W_d = \emptyset$
  for i ∈ [1,s]: **do**
    $v \sim \mathcal{N}(0, \mathbb{I}) \in R^K$ ▷ sample v
    $w_g = v - \langle v, g_u\rangle g_u$ ▷ no high-change…
    $w_h := w_g - \sum_{i=1}^{k}\langle w_g, U_i^{\downarrow}(H)\rangle w_g$ ▷ …dirs
    $\hat{W}_i = D(l + \beta w_h)$
    $W_d \leftarrow W_d \cup \hat{W}_i$ ▷ add to candidates
  end for
  $W_L* = \text{MMR}(W, R, W_d)$ ▷ prune candidates
  $W_{\text{Projected}} = \arg\min f(W_L)$ ▷ project
  $R \leftarrow R \cup W_{\text{Projected}}$ ▷ augment training set
end procedure

(i) Deformation-aware perturbation in latent space. We devise a method to perturb a given code in latent space without modifying its deformation energy significantly: let W be a deformation, and $l \sim E(W) \in R^K$ a latent code achieved from encoding it. We aim to find the low-energy perturbation modes. To that end, we denote the (normalized) energy's gradient with respect to l by,

$$g_u := \nabla_l f(D(l))/\|\nabla_l f(D(l))\|. \tag{5}$$

Similarly, let H denote the Hessian of the deformation energy with respect to the latent code,

$$H := \nabla_l \nabla_l f(D(l)). \tag{6}$$

To get an (arbitrary) perturbation a random vector $v \in R^K$ from a normal distribution, we first eliminate the gradient direction (direction of maximal energy change) from the perturbation via projection:

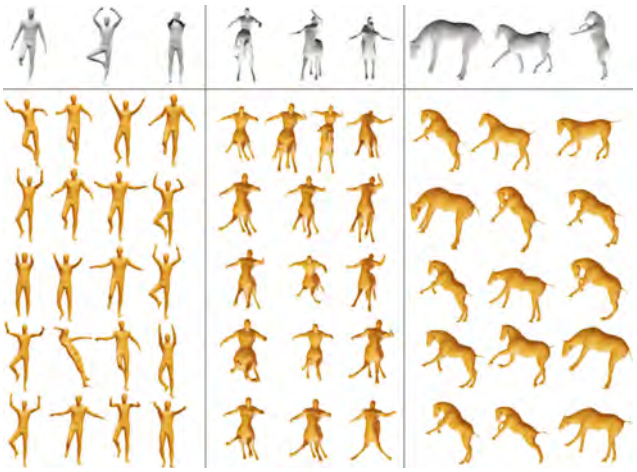

Figure 4: Training GLASS on the human, centaur, and horse meshes using the 3 examples each (top). (Bottom) We show random samples from the latent space, which combine different properties learned from the example deformations.

$$w_g := v - \langle v, g_u \rangle g_u, \tag{7}$$

where $\langle . \rangle$ represents the inner product. Next, we remove from w_g the eigenvectors of H corresponding to the highest eigenvalues of the Hessian of f. Let $U_i^{\downarrow}(H)$ denote its eigenvectors, ordered in *descending* order w.r.t their corresponding eigenvalues (i.e., highest eigenvalues come first).

$$w_h := w_g - \sum_{i=1}^{k} \langle w_g, U_i^{\downarrow}(H) \rangle w_g. \tag{8}$$

Finally, using the projected perturbation w_h we perturb the latent code to get $\tilde{l} \leftarrow l + \beta w_h$ where β is the step size. We repeat this process s times from the same latent code l with random v's to get s perturbed codes $\tilde{l}^1, \tilde{l}^2, \ldots \tilde{l}^s$. Let $\{\tilde{W}^1, \tilde{W}^2 \ldots \tilde{W}^s\}$ denote the decoded perturbed deformations where $\tilde{W}^j = D(\tilde{l}^j)$.

In our implementation, we use Pytorch's GPU version of SVD to compute the eigenvectors of H. This implementation can fail to find a decomposition, in which case we will only eliminate the gradient direction from v, and update the latent code with $l \leftarrow l + \beta w_g$.

(ii) Data-driven pruning of the perturbed deformations. We select a single deformation from the candidate deformations $\{\tilde{W}^j\}$ to be added to the dataset, via the Maximal Marginal Relevance (MMR) ranking [7]. Specifically, MMR gives a higher score to perturbations that are similar to the unperturbed W, but different from the entire deformations set R. We compute this as,

$$F(w) = \alpha M(w, W) - (1 - \alpha) \max_{r \in R} M(w, r). \tag{9}$$

where $M(x, y)$ is the cosine similarity between x and y. We choose the deformation $W^* \in W_d$ that maximizes the MMR function with its latent code denoted l^*.

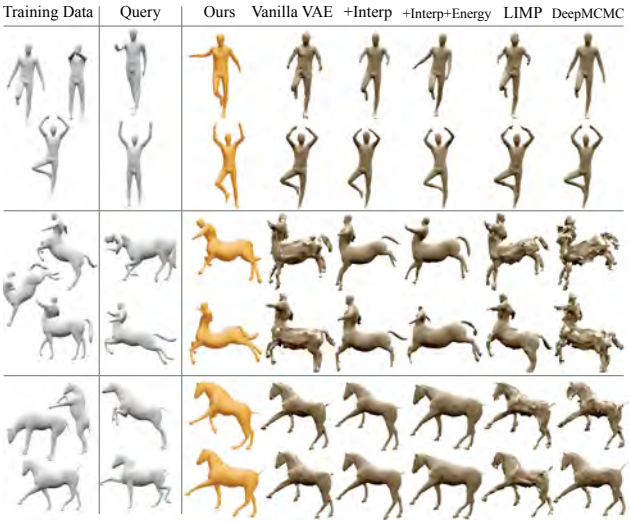

Figure 5: *Generation results evaluated by coverage.* We train different methods on the same training data (col 1) and generate comparable numbers of shapes. Given two shapes from the holdout data (col 2), we evaluate the methods by finding the closest generated shape (cols 3-8). Note how baseline techniques exhibit strong artifacts and usually do not generate a good match to the query shape.

(iii) Deformation-aware projection to smooth, low-energy deformations. Although the deformation-aware perturbation somewhat avoids high-energy states, the perturbed deformation W^* may still exhibit undesirable artifacts such as lack of smoothness or high deformation energy. Hence, we project the code to have lower deformation energy *with respect to the unperturbed* W. We achieve this by treating W now as the rest pose, defining a deformation energy with respect to it, f_W. We perform a fixed number of gradient descent steps starting from W^* to lower the energy, which yields the final deformation $W_{\text{Projected}}$. In our experiments, we optimize up to the threshold of 10^{-5}. We append this to the current (training) data set.

4. Experiments

We evaluate GLASS on public data of humans (FAUST [4]) and creatures (TOSCA [6]) in different poses. In our experiments, we sample L landmark poses of a model from a dataset and train our method. We evaluate quality and diversity of newly generated poses as well as interpolation sequences between both landmark and generated poses. We denote our experiments as "SubjectName-X" to indicate the type of the subject and the number of landmark poses provided as an input to our method; most results use between 3 and 10 landmarks.

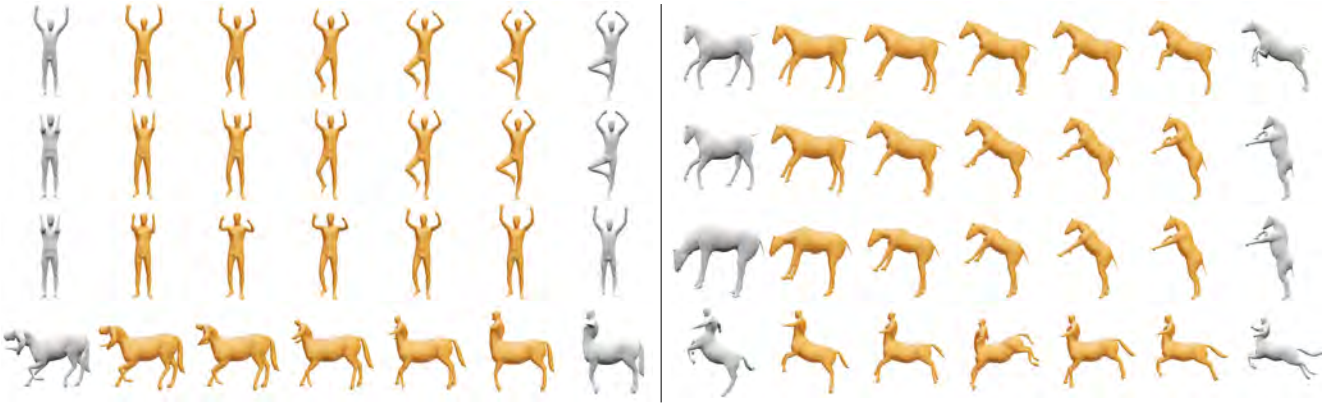

Figure 6: *Interpolation results.* In gray, we show two landmark shapes. In gold, we show the decoded meshes after we linearly interpolate the latent space between these two landmarks. All models are trained on only 5 landmarks.

4.1. Evaluation Metrics

The goal of a data augmentation method is to generate natural, smoothly-varying and diverse deformations. To that end we propose three metrics to evaluate performance.

(i) Coverage: While it is difficult to evaluate whether generated poses are meaningful and diverse, we propose to use the holdout data (S_H) that was not part of the input exemplars to see if the newly generated poses L_G cover every holdout example:

$$M_{\text{coverage}} := \sum_{s \in S_H} \min_{g \in L_G} D(s,g)/|S_H|,$$

where $D(s,g)$ is the average Euclidean distance between corresponding vertices of shapes s and g.
(ii) Smoothness: This metric measures how well a method preserves the original intrinsic structure of the mesh. We compute the mean curvature obtained from the discrete Laplace-Beltrami operator:

$$M_{\text{smoothness}} := \sum_{i=1}^{N} \frac{1}{2} \|\Delta(V_i)\|$$

where Δ is the area-normalized cotangent Laplace-Beltrami operator and N is the number of mesh vertices.
(iii) Interpolation smoothness: In addition to measuring quality of individual meshes, we also evaluate the quality of interpolations between pairs of shapes. Since none of the existing methods guarantee a clear relationship between the distances in the latent space and differences between frames, we first densely sample 1000 poses between pairs of landmark deformations and then keep a subset of 30 poses so that they have approximately equal average Euclidean distance between subsequent frames. We then measure the standard deviation of these Euclidean distances, as a way to penalize interpolations that yield significant jumps between frames. While this metric is imperfect (e.g., variance could decrease as we increase the sampling rate), we found it to stabilize in practice after 1000 poses (we sampled up to 3000), which suggests that denser sampling would not reveal new poses in the latent space.

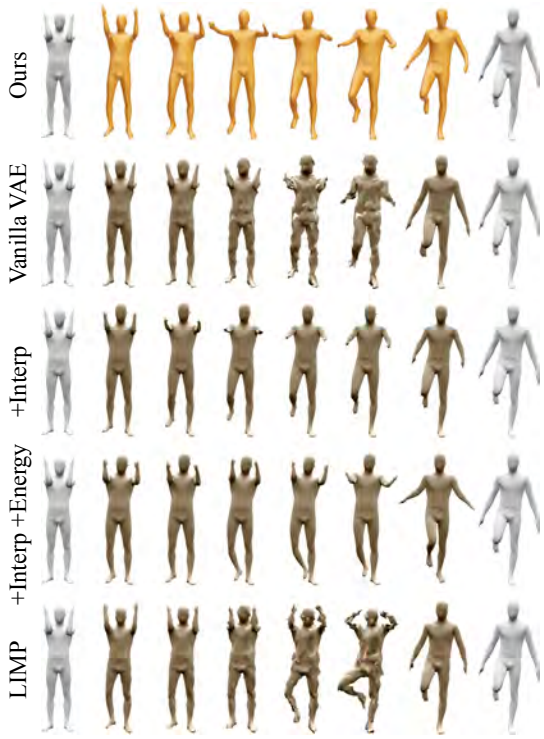

Figure 7: We compare the interpolation results between our method, several ablations of our method, and prior work.

4.2. Comparison to Alternative Methods

While existing methods are not designed to learn generative latent spaces from very sparse data, we adapt them as baselines.

Table 1: Surface smoothness and coverage with respect to excluded set, of generated samples. Lower is better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	Deep-MCMC	GLASS
Faust-3	1.00 / 0.08	0.66 / 0.08	0.71 / 0.08	0.96 / 0.08	0.98 / 0.07	**0.59 / 0.05**
Faust-5	1.00 / 0.07	0.69 / 0.08	0.69 / 0.07	1.04 / 0.08	1.02 / 0.08	**0.62 / 0.06**
Faust-7	1.00 / 0.04	0.74 / 0.04	0.66 / 0.05	1.07 / 0.04	0.95 / 0.04	**0.59 / 0.03**
Centaurs-3	1.00 / 0.11	0.77 / 0.09	0.72 / 0.09	0.96 / 0.11	1.03 / 0.11	**0.68 / 0.06**
Centaurs-4	1.00 / 0.10	0.70 / 0.10	0.70 / 0.10	0.99 / 0.10	0.96 / 0.10	**0.69 / 0.07**
Horses-3	1.00 / 0.06	0.74 / 0.07	0.69 / 0.08	1.14 / 0.06	0.95 / 0.06	**0.65 / 0.05**
Horses-4	1.00 / 0.06	0.70 / 0.06	0.72 / 0.06	1.05 / 0.06	1.07 / 0.05	**0.69 / 0.04**

1. *Vanilla VAE:* We train a VAE using only the sparse set of shapes, with no data augmentation, using the encoder and decoder detailed in Section 3.2.

2. *+Interpolation:* We generate new shapes by interpolating between all pairs of available landmarks by simply averaging coordinates of corresponding vertices. We interpolate such that the amount of augmented data is equivalent to what is generated with our method (2500 shapes). Then we train a VAE with these poses.

3. *+Interp. +Energy:* This is an extension of the previous method. Since raw interpolation can deviate from the underlying shape space, in addition to interpolation, we perform projection by minimizing the sum of ARAP energies with respect to both shapes in the pair.

4. *LIMP-ARAP [10]:* This method is motivated by the training strategy proposed in LIMP [10]. They train a VAE with pairs of shapes in every iteration - for each pair, they pick a random latent code on the line between the two, decode it to a new shape, and minimize its energy. Since we only want to compare augmentation strategies we adapt LIMP to use ARAP energy.

5. *Deep-MCMC [29]:* This method explores parameter variations via a latent space. They encode a given dataset and generate samples by performing HMC steps in the latent space and decoding the generated codes. This method is not suitable for interpolation, so we only evaluate its ability to generate novel poses.

Generation Experiments We next evaluate the quality of shapes generated with our method. After training, we sample latent codes from a Unit Gaussian in R^K, and decode with our decoder to generate samples (see Figures 1 and 4). See how generated poses look substantially different from the training data and combine features from multiple input examples. Our method can also be used in incremental training, where we add new samples to previously-trained latent space to discover new poses (see supplemental).

We compare our approach to all the baselines. We sample from Unit Gaussian for all VAE-based techniques, where the only exception is Deep-MCMC where we use latent-space HMC as proposed in their work. We show qualitative results in Figure 5 and quantitative evaluations in Table 1. Each cell reports smoothness and coverage errors, normalized based on the corresponding errors for Vanilla-VAE. Note that our method outperforms all baselines in its ability to generate novel and plausible poses (i.e., the poses from the hold-out set of the true poses).

Interpolation Experiments We compare our method and the first four baselines by evaluating the quality of interpolations produced between all pairs of landmark shapes (we omit Deep-MCMC since it is not suitable for interpolation). We show our results in Figure 6 and comparisons in Figure 7 with corresponding stats in Table 3. Each cell reports smoothness, ARAP score, and interpolation quality, and to make results more readable we normalize the scores using the corresponding value for Vanilla VAE.

Our method performs the best with respect to ARAP score and also yields consistently smoother shapes with fewer artifacts, both in terms of individual surfaces, as well as discontinuities in interpolation sequences. LIMP-ARAP is performing consistently worse than all baselines except Vanilla-VAE. Interpolation-based baselines sometimes yield smoother interpolations, something we would expect to be true for simpler, linear motions. However, as we will demonstrate they are limited in their ability to synthesize novel plausible poses.

Table 2: Correspondence error on the Faust INTRA benchmark, by GLASS-augmenting 3D-CODED with deformations sampled from our method.

Data	+GLASS augmentation	Error (cm)
Faust-3	0	26.90
Faust-3	3,065	**12.06**
Faust-7	0	22.10
Faust-7	3,573	**13.95**
SMPL-650	0	24.69
SMPL-650	84,000	5.41
SMPL-650	192,000	**4.37**

Table 3: Surface smoothness, ARAP energy, and standard deviation of inter-frame spacing between landmarks by interpolation across different datasets. All results are normalized such that Vanilla VAE is 1.0, and lower numbers are better.

Data	Vanilla VAE	+Interpolation	+Interp. +Energy	LIMP	GLASS
Faust-3	1.00 / 1.00 / 1.00	0.62 / 0.36 / 0.59	0.62 / 0.36 / 0.35	1.06 / 1.34 / 0.40	**0.58 / 0.26 / 0.31**
Faust-5	1.00 / 1.00 / 1.00	0.61 / 0.34 / 0.79	0.62 / 0.32 / **0.35**	1.04 / 1.18 / 1.01	**0.58 / 0.22** / 0.62
Faust-7	1.00 / 1.00 / 1.00	0.63 / 0.33 / 0.73	0.63 / 0.34 / **0.13**	0.93 / 0.97 / 0.85	**0.62 / 0.31** / 0.71
Faust-10	1.00 / 1.00 / 1.00	0.66 / 0.33 / 0.30	0.67 / 0.37 / 0.27	0.88 / 0.76 / 0.36	**0.61 / 0.26 / 0.25**
Centaurs-3	1.00 / 1.00 / 1.00	0.63 / 0.35 / 0.57	0.66 / 0.36 / 0.43	0.89 / 1.03 / 0.79	**0.57 / 0.22 / 0.40**
Centaurs-4	1.00 / 1.00 / 1.00	0.63 / 0.23 / 0.65	0.66 / 0.25 / **0.42**	1.00 / 1.04 / 0.78	**0.61 / 0.20** / 0.70
Centaurs-6	1.00 / 1.00 / 1.00	0.67 / 0.38 / 0.73	0.67 / 0.39 / 0.66	0.93 / 0.87 / 0.87	**0.65 / 0.34 / 0.60**
Horses-3	1.00 / 1.00 / 1.00	0.58 / 0.41 / 0.62	0.63 / 0.56 / **0.42**	1.03 / 1.31 / 0.96	**0.55 / 0.35** / 0.64
Horses-4	1.00 / 1.00 / 1.00	0.56 / 0.31 / 0.48	0.56 / 0.42 / 0.53	0.93 / 0.98 / 0.60	**0.54 / 0.28 / 0.42**
Horses-8	1.00 / 1.00 / 1.00	0.65 / 0.38 / **0.30**	0.64 / 0.39 / 1.03	0.89 / 0.84 / 0.70	**0.63 / 0.35** / 0.76

Table 4: This Table presents ablations on various steps of our method with respect to interpolation evaluation metrics. Starting with vanilla VAE (1) we add deformation energy (2), deformation-aware perturbations (3,4), and projection (5,6) steps.

Data	1: Vanilla VAE	2: (1)+L_{deform}	3: (1) + perturb	4: (2) + perturb	5: (3)+project	6: (4)+project
Faust-10	1.00 / 1.00 / 1.00	0.96 / 0.91 / 0.86	0.76 / 0.52 / 0.39	0.72 / 0.48 / 0.34	0.62 / 0.27 / 0.32	**0.61 / 0.26 / 0.25**
Centaurs-6	1.00 / 1.00 / 1.00	0.95 / 0.87 / 0.93	0.73 / 0.61 / 0.82	0.68 / 0.47 / 0.73	0.67 / 0.37 / 0.61	**0.65 / 0.34 / 0.60**
Horses-8	1.00 / 1.00 / 1.00	0.97 / 0.98 / 0.91	0.73 / 0.84 / 0.84	0.71 / 0.50 / 0.80	0.64 / 0.37 / 0.77	**0.63 / 0.35 / 0.76**

4.3. Using GLASS for Learning Correspondences

We now evaluate our data augmentation technique on the practical task of learning 3D correspondences between shapes. We pick a state-of-the-art correspondence learning method, 3D-CODED [15], as a reference. Originally, this method was trained on 230k deformations, most of them sampled from the SMPL model [26] and 30k synthetic augmentations. Obtaining this large training dataset is a challenge, so we evaluate how well this method could perform with a smaller training set, with and without the augmentation proposed in this paper.

We train 3D-CODED using different small datasets that are augmented with a number of additional deformations sampled from our model. We show the results of this experiment in Table 2. We see that our method consistently provides a significant improvement over training 3D-CODED with the original landmarks and that additional samples continue to improve the results on the SMPL-650 dataset. For a reference, the correspondence error of 3D-CODED trained on the full 230k pose dataset is 1.98cm.

4.4. Method Ablations

In this section we evaluate the contribution of various steps used in GLASS with respect to interpolation metrics. We evaluate on the Faust-10, Centaurs-6 and Horses-8 datasets. Since these are all the available data for them, there is no hold-out set, so we do not measure coverage. Starting with Vanilla-VAE (that only uses $L_{\text{Reconstruction}}$ and L_{Gaussian}), we first add the deformation energy loss ($L_{\text{Deformation}}$). Table 4(1,2) shows this improves all metrics.

Next, we consider our perturbation strategy (Section 3.3 i, ii) and add it to both vanilla as well as energy-guided VAE (Table 4.3, 4.4). Note that in both of these cases, our perturbation strategy improves on all metrics by up to 1.5x. We observe that $L_{\text{Deformation}}$ performs better because it makes the latent-space conducive for sampling low energy shapes. $L_{\text{Deformation}}$ helps our perturbation strategy find low energy shapes that are suitable for our subsequent projection step. Due to this,we discover shapes with energy as low as 0.001, while without it, the discovered shapes can have energy > 0.1. This difference helps the subsequent projection step converge faster to our required threshold of 10^{-5}.

Finally, we look at the projection step (Section 3.3 iii). We add it to both baseline techniques that have perturbation, and report results in Table 4.5, 4.6, where column (6) corresponds to our final method. Adding the projection step improves the smoothness and ARAP scores by up to 1.3x. After projection, our shapes have very low ARAP, in the order of 10^{-5}. Since these are added back to the training set, we observe that perturbation steps in future iterations find lower energy shapes. This further improves convergence of the projection step in future iterations. To summarize, $L_{\text{Deformation}}$ helps both the perturbation and projection steps converge faster to low energy shapes, and since projected shapes are encoded again by training, both perturbation and projection steps require fewer iterations.

5. Conclusion

GLASS is shown to be an effective generative technique for 3D shape deformations, relying solely on a mere handful of examples, and a given deformation energy. The main limitation of our method is its reliance on a given mesh with vertex correspondences, preventing its use on examples with different triangulations, and we set the goal of generalizing it to arbitrary geometries as important future work.

We believe our proposed technique opens many future directions. There are many other deformation energies that could be explored; e.g., densely sampling conformal (or quasi-conformal) deformations from a given sparse set can be an extremely interesting followup. More broadly, replacing the deformation energy with learned energies, such as the output of an image-discriminator, may enable generating plausible images, given a very sparse set of examples.

References

[1] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. In *SIGGRAPH*, 2003.

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape completion and animation of people. In *SIGGRAPH*, 2005.

[3] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999.

[4] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*, 2014.

[5] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *TVCG*, 14(1), 2008.

[6] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[7] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA, 1998. Association for Computing Machinery.

[8] Amit Chaudhary. A visual survey of data augmentation in NLP, 2020. `https://amitness.com/2020/05/data-augmentation-for-nlp`.

[9] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3D structures. *Comput. Graph. For. (Eurographics STAR)*, 2020.

[10] Luca Cosmo, Antonio Norelli, Oshri Halimi, Ron Kimmel, and Emanuele Rodolà. LIMP: Learning latent shape representations with metric preservation priors. *ECCV*, 2020.

[11] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Měch, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In *CVPR*, 2020.

[12] Lin Gao, Yu-Kun Lai, Jie Yang, Ling-Xiao Zhang, Shihong Xia, and Leif Kobbelt. Sparse data driven mesh deformation. *TVCG*, 27(3), 2021.

[13] 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 Trans. Graph.*, 38(6), 2019.

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[15] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3D-CODED: 3D correspondences by deep deformation. *ECCV*, 2018.

[16] Frédéric Hélein and John C. Wood. *Handbook of Global Analysis*, chapter Harmonic Maps. 2008.

[17] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. *Computer Graphics Forum*, 28(2), 2009.

[18] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4), 2011.

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

[20] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *SIGGRAPH*, 2005.

[21] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3), 2007.

[22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[23] Hamid Laga. A survey on non-rigid 3D shape analysis. *CoRR*, abs/1812.10111, 2018.

[24] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH*, 2002.

[25] J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. *Eurographics State of the Art Reports*, 2014.

[26] M. Loper, N. Mahmood, J. Romero, Pons-Moll, and M. J. G., Black. Smpl: A skinned multi-person linear model. In *SIGGRAPH Asia*, 2015.

[27] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3D faces using convolutional mesh autoencoders. In *ECCV*, 2018.

[28] Guodong Rong, Yan Cao, and Xiaohu Guo. Spectral mesh deformation. *The Visual Computer*, 24, 2008.

[29] Babak Shahbaba, Luis Martinez Lomeli, Tian Chen, and Shiwei Lan. Deep Markov chain Monte Carlo. *CoRR*, abs/1910.05692, 2019.

[30] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.

[31] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 2019.

[32] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP*, 2007.

[33] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3D mesh models. In *CVPR*, 2018.

[34] Qingyang Tan, Zherong Pan, Lin Gao, and Dinesh Manocha. Realtime simulation of thin-shell deformable materials using CNN-based mesh embedding. *IEEE Robotics and Automation Letters*, 5(2), 2020.

[35] Christoph Von-Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. Real-time nonlinear shape interpolation. *ACM Trans. Graph.*, 34(3), 2015.

[36] Yifan Wang, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020.

[37] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *CoRR*, abs/2002.12478, 2021.

[38] Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. A revisit of shape editing techniques: from the geometric to the neural viewpoint. *CoRR*, abs/2103.01694, 2021.

[39] Mehmet Ersin Yumer and Levent Burak Kara. Co-constrained handles for deformation in shape collections. *ACM Trans. Graph.*, 33(6), 2014.

Supplementary for GLASS: Geometric Latent Augmentation for Shape Spaces

6. Dataset

Each of the triangle meshes used for evaluation contains 1000 vertices and 1996 faces. We report sample generation on higher resolution of 6890 vertices in a later section in this supplementary. SMPL-650 contains only 650 shapes sampled from SMPL.

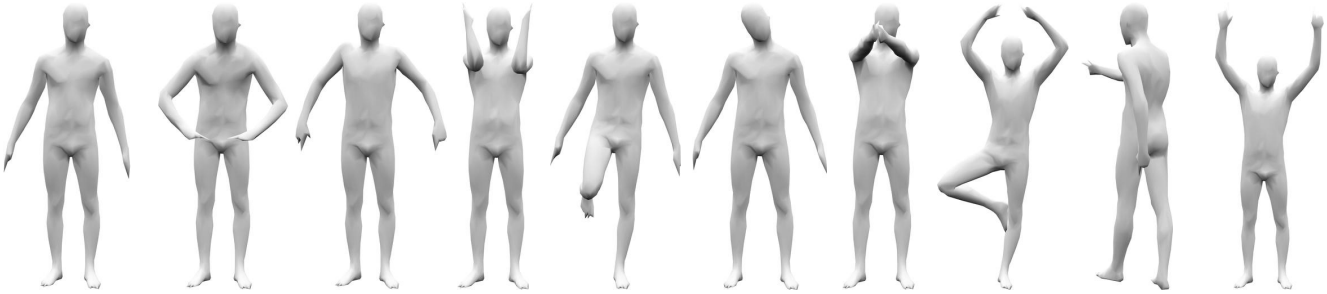

Figure 8: Faust-10 dataset, with all the 10 poses available in Faust. All models are consistently scaled – the models may look different in the following images due to image-specific scaling.

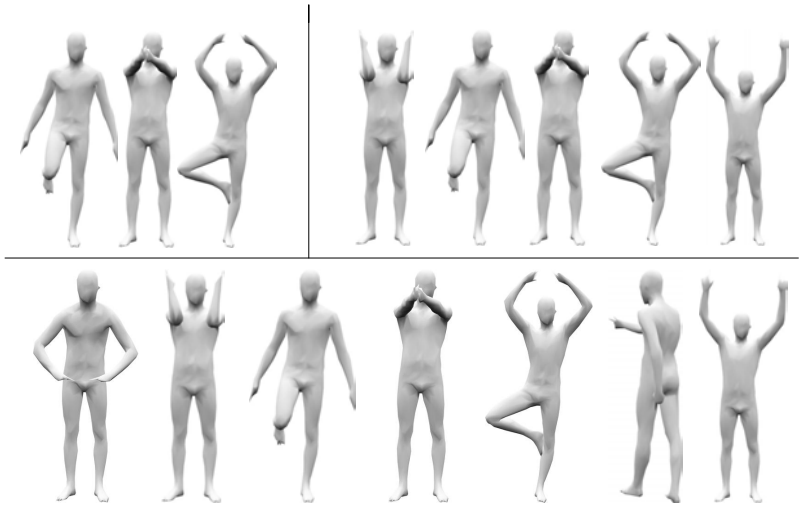

Figure 9: Faust-3 (top-left), Faust-5 (top-right) and Faust-7 (bottom) sampled from Faust-10.

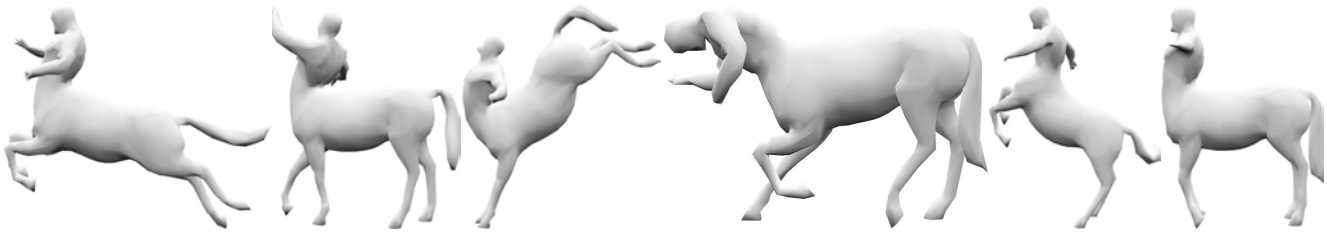

Figure 10: Centaurs-6 dataset, with all the 6 poses available in the Tosca dataset.

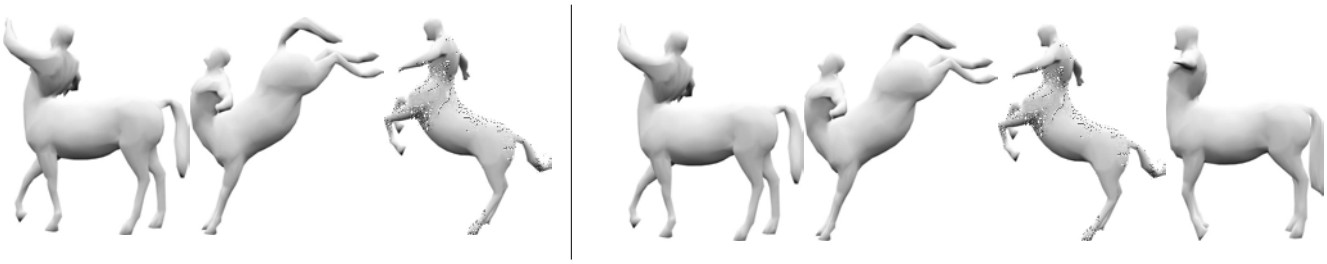

Figure 11: Centaurs-3 (left) and Centaurs-4 (right), sampled from Centaurs-6.

Figure 12: Horses-8 dataset, with all the 8 poses available in Tosca.

Figure 13: Horses-3 (left) and Horses-4 (right), sampled from Horses-8.

7. Interpolation frames

(i) Some interpolations between pairs (shown in gray) of Faust shapes.

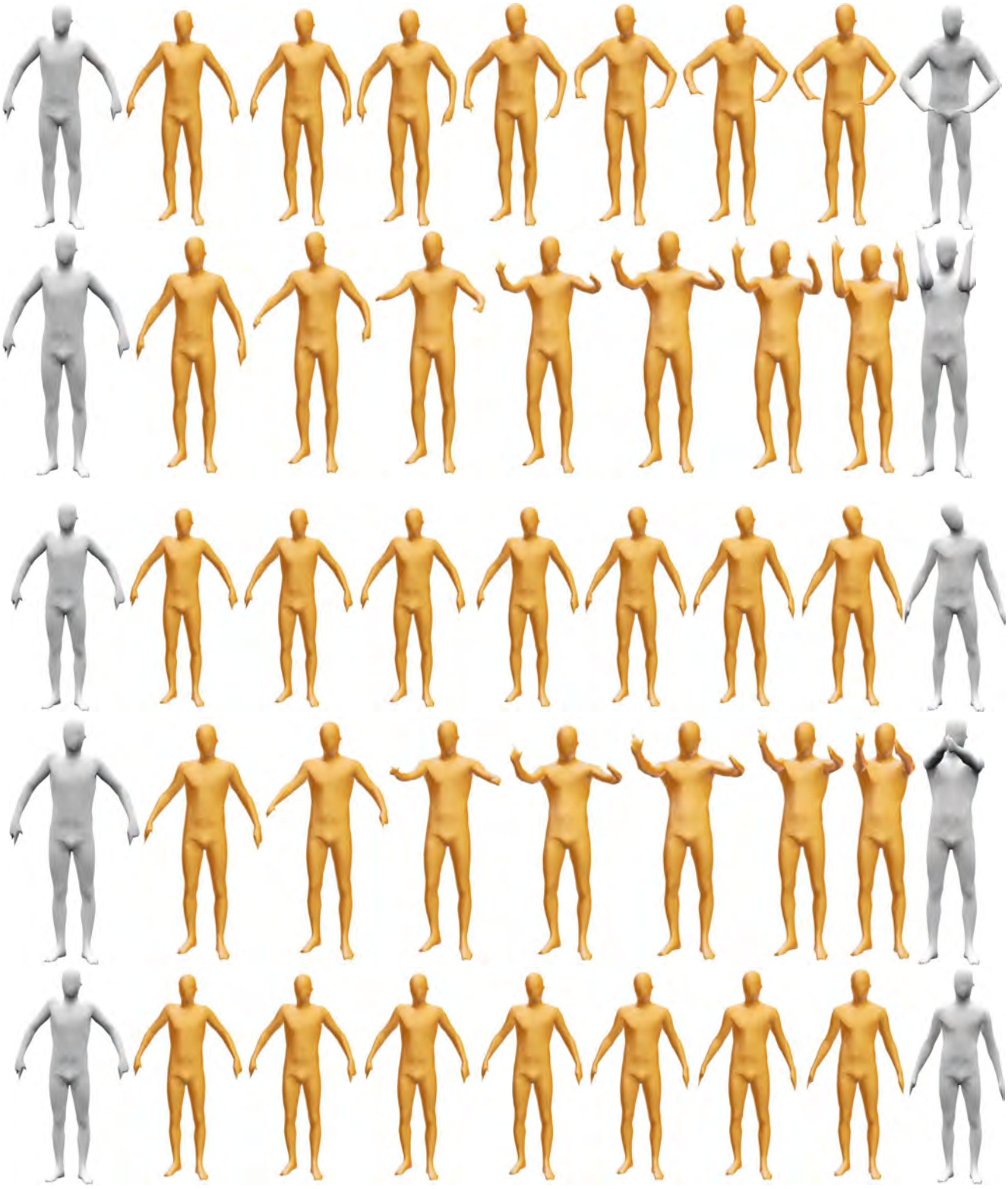

Figure 14: Faust-10 Interpolation results 1/6.

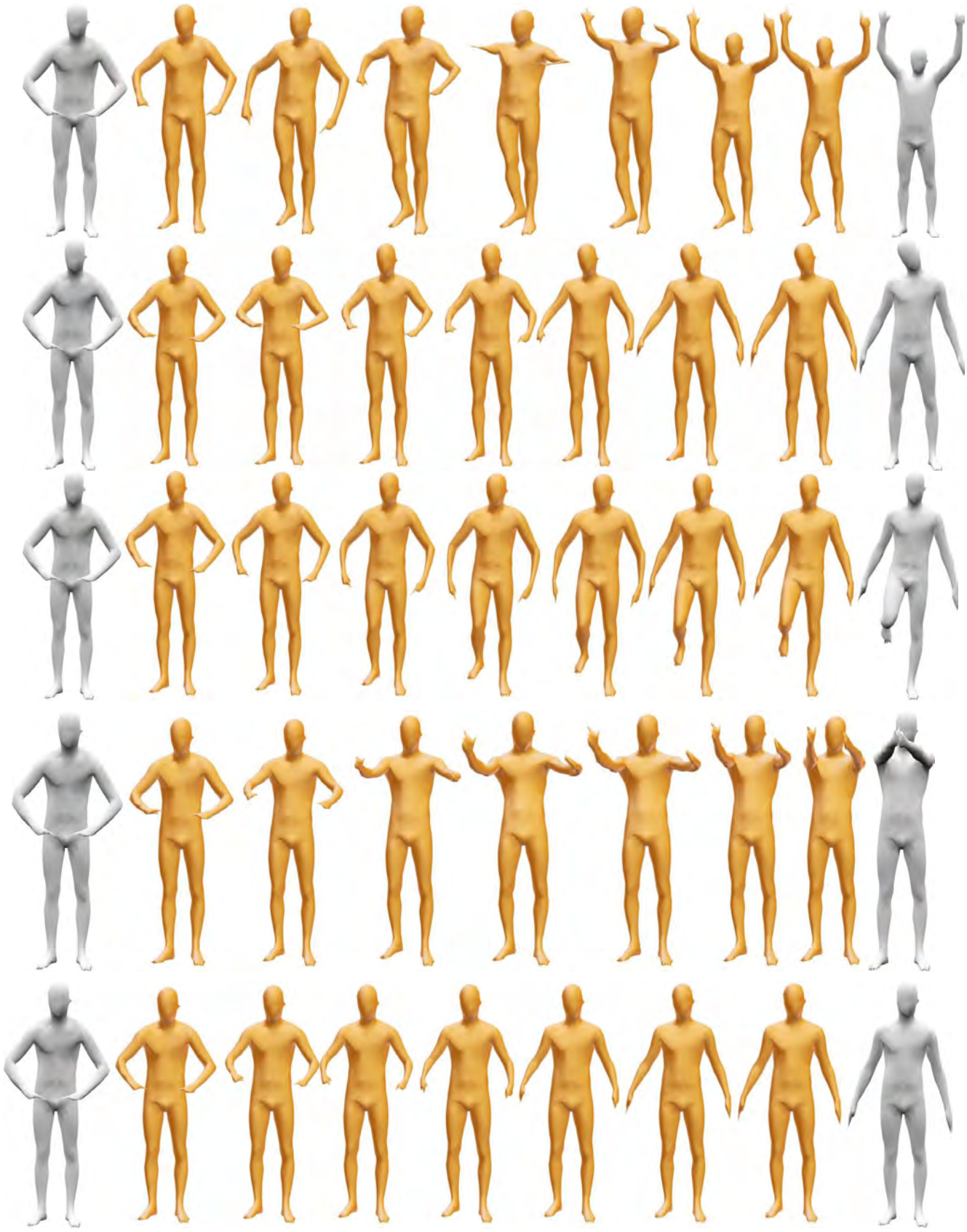

Figure 15: Faust-10 Interpolation results 2/6.

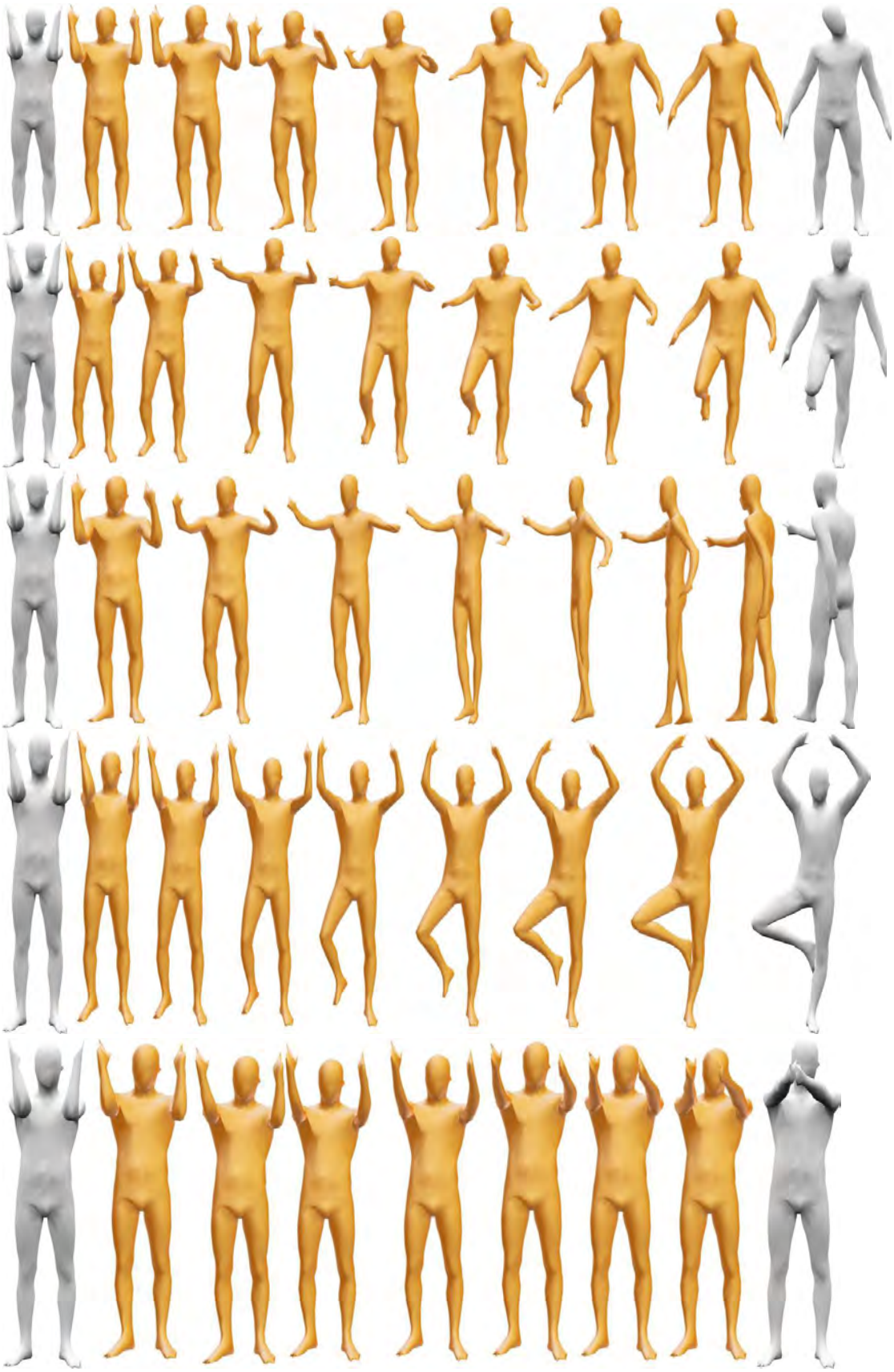

Figure 16: Faust-10 Interpolation results 3/6.

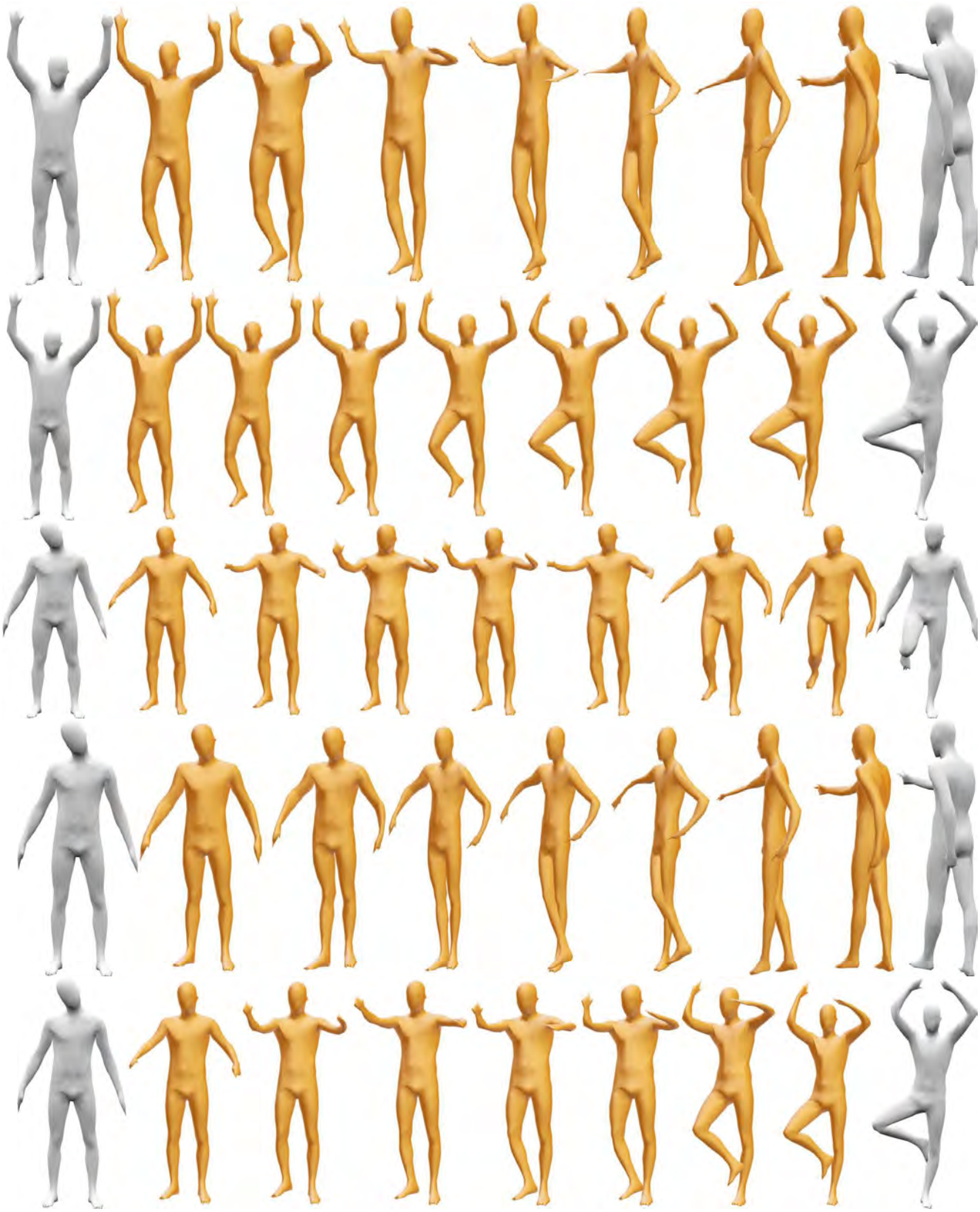

Figure 17: Faust-10 Interpolation results 4/6.

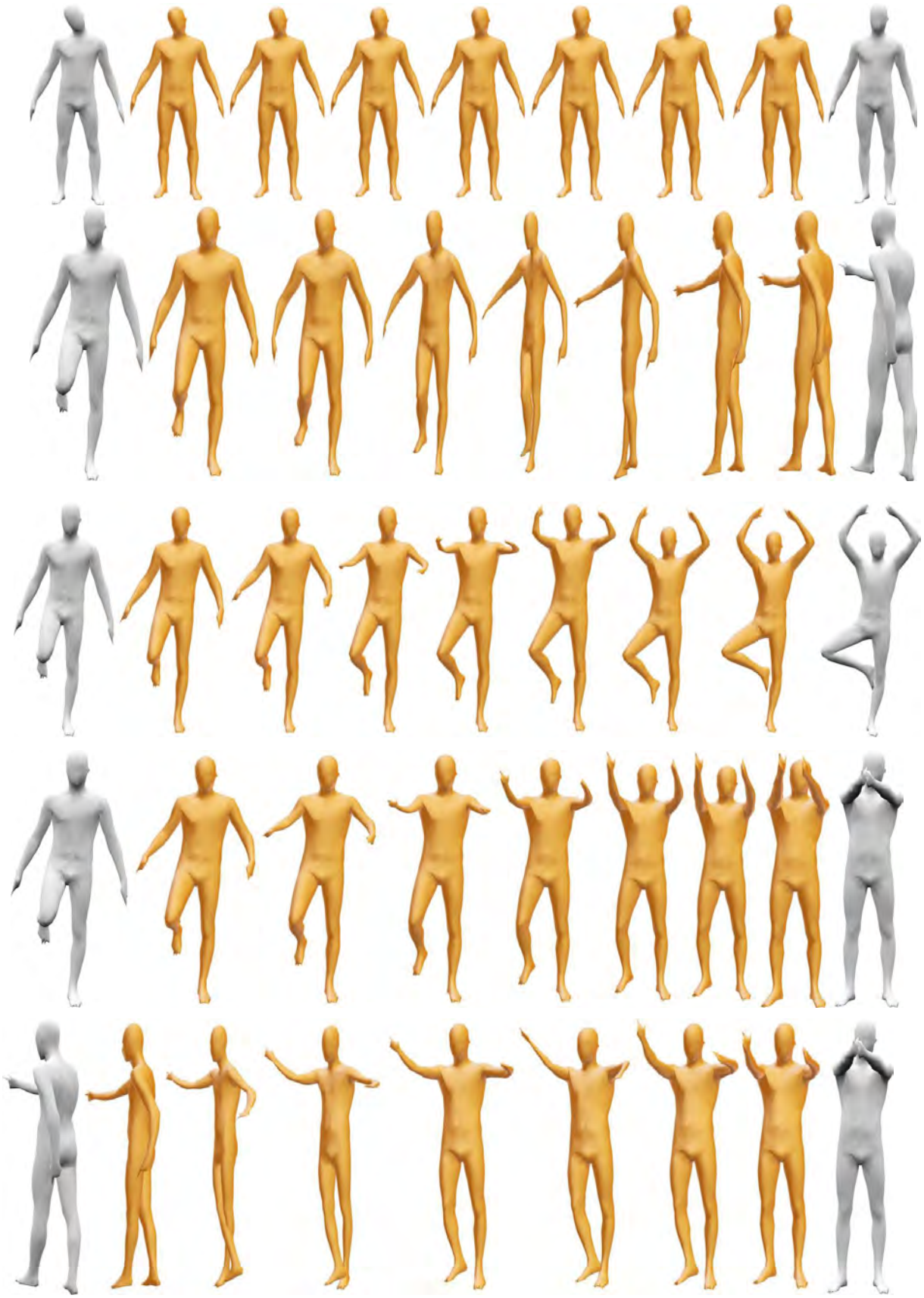

Figure 18: Faust-10 Interpolation results 5/6.

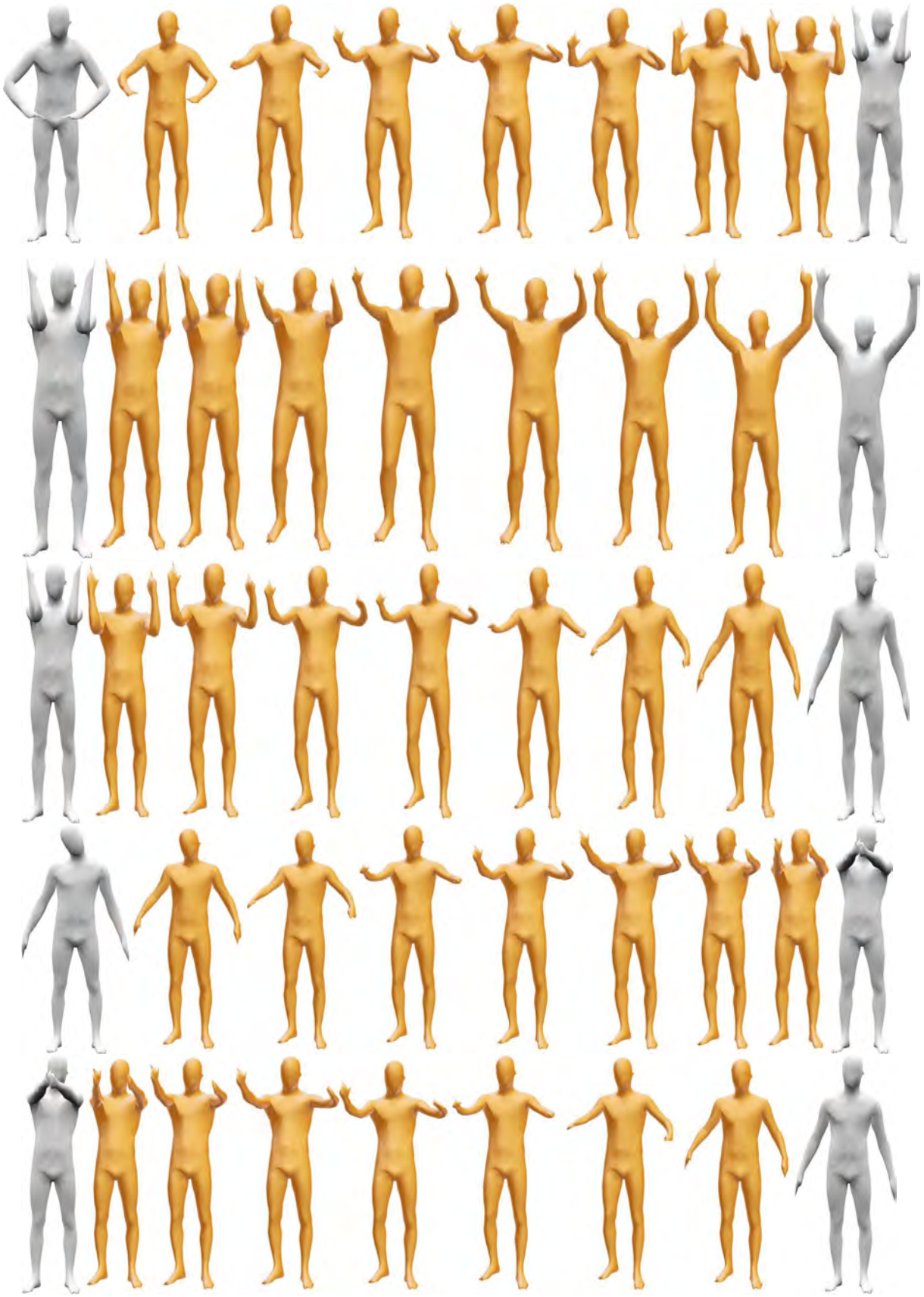

Figure 19: Faust-10 Interpolation results 6/6.

(ii) Some interpolations between pairs (shown in gray) of Tosca Centaurs.

Figure 20: Centaurs Interpolation results 1/3.

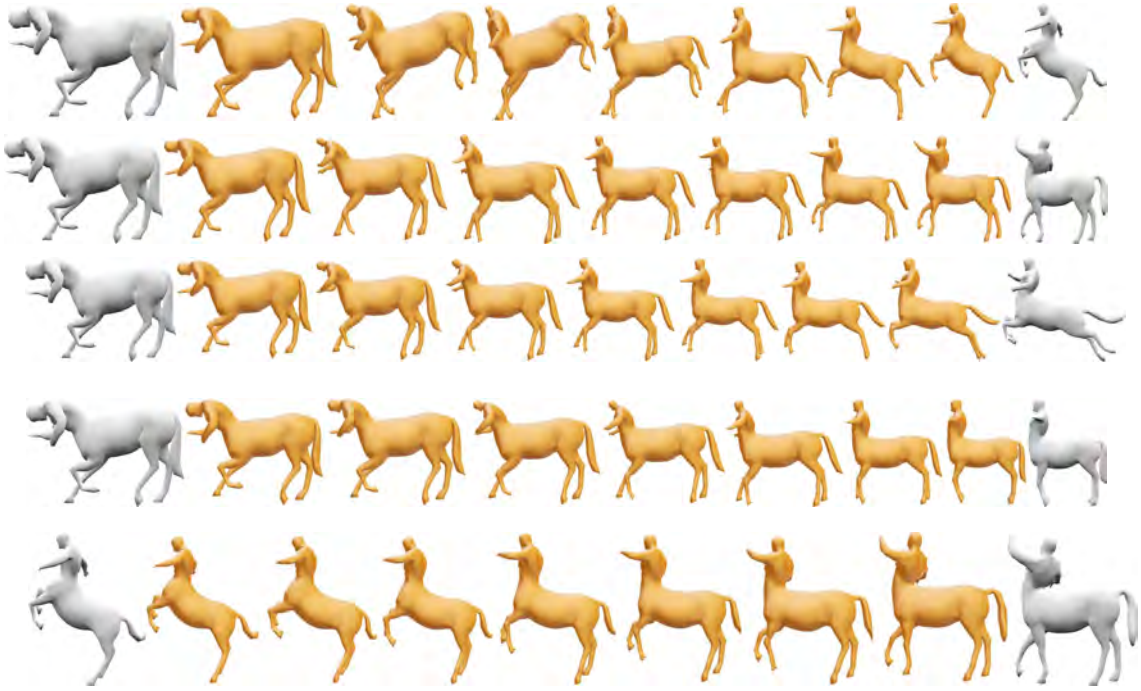

Figure 21: Centaurs Interpolation results 2/3.

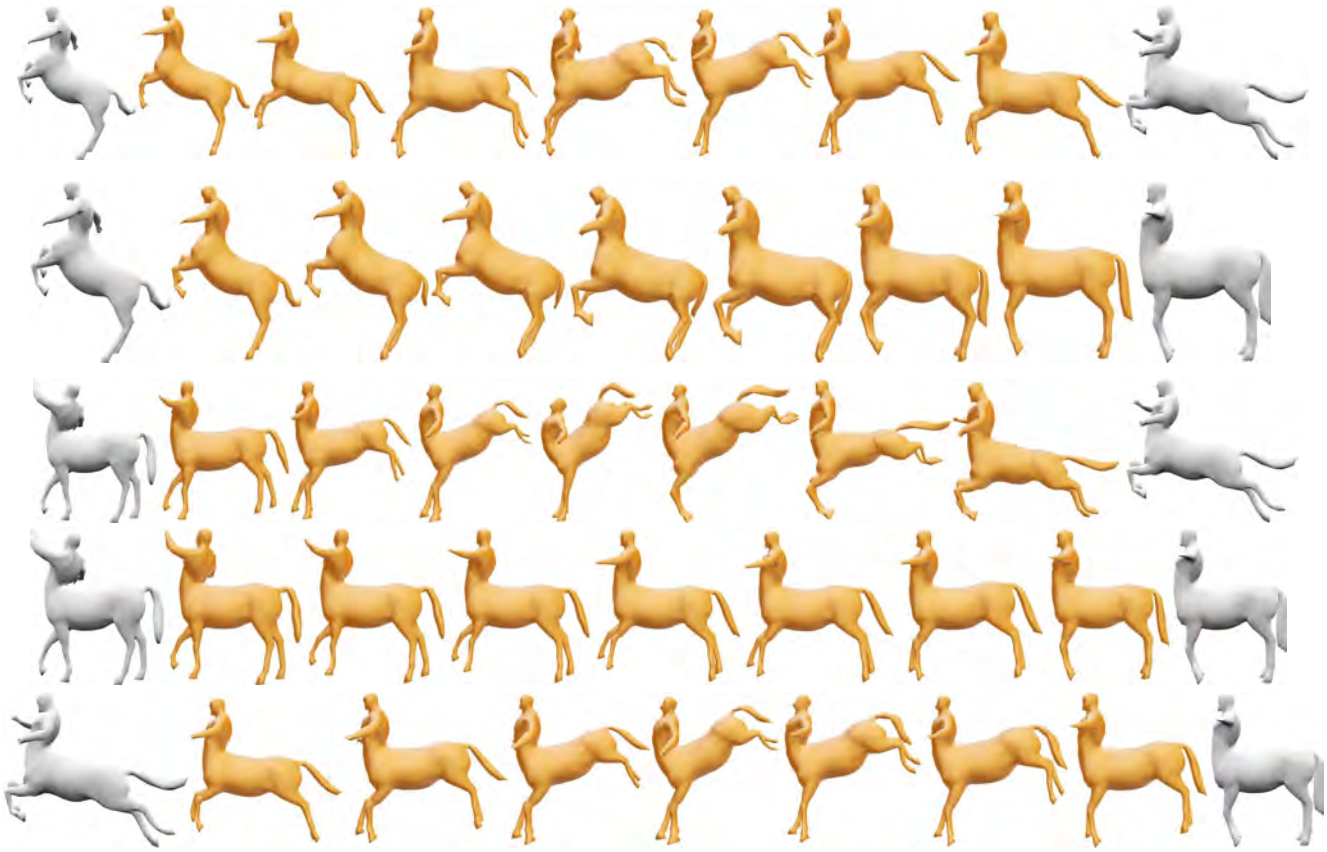

Figure 22: Centaurs Interpolation results 3/3.

(iii) Some interpolations between pairs (shown in gray) of Tosca Horses.

Figure 23: Horses Interpolation results 1/6.

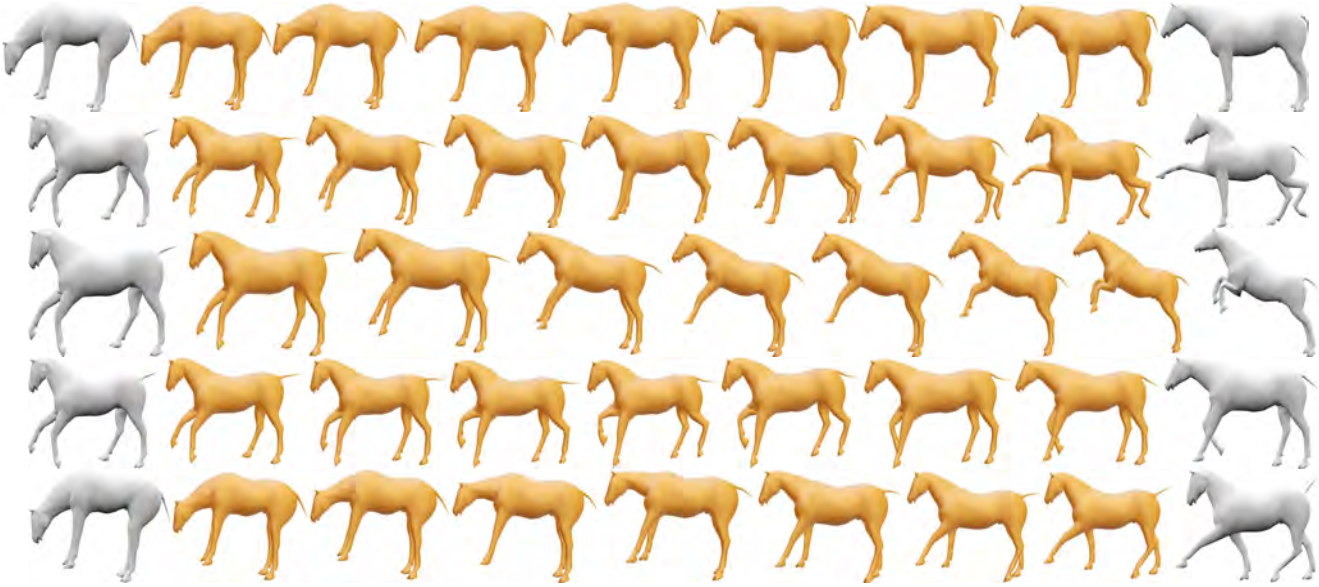

Figure 24: Horses Interpolation results 2/6.

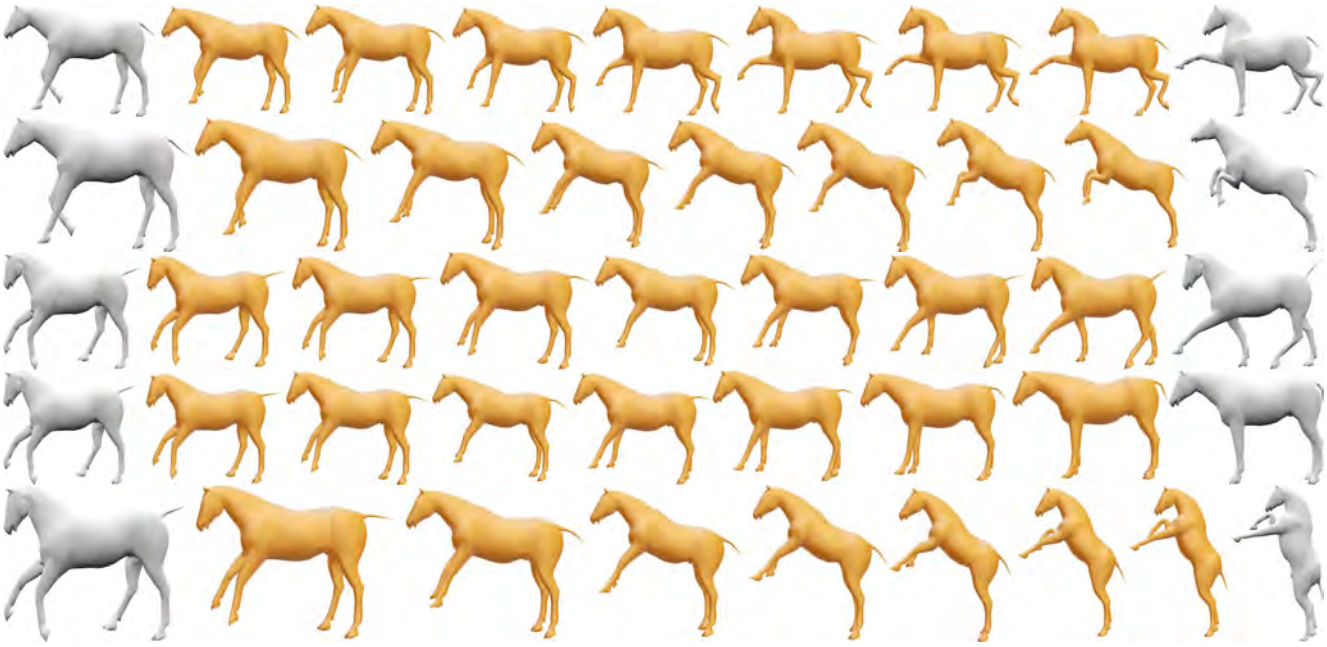

Figure 25: Horses Interpolation results 3/6.

Figure 26: Horses Interpolation results 4/6.

Figure 27: Horses Interpolation results 5/6.

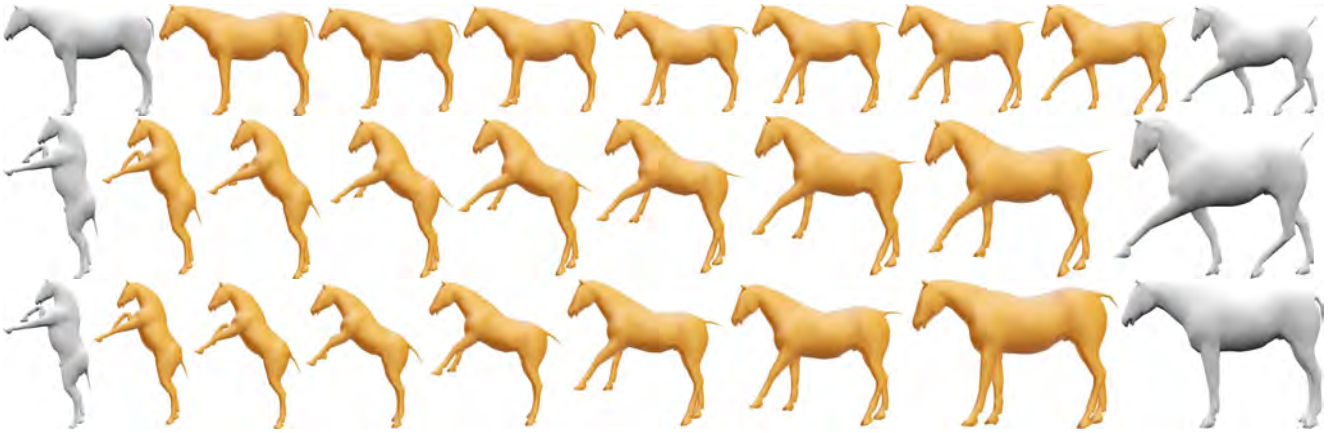

Figure 28: Horses Interpolation results 6/6.

8. Faust-10 Coverage

We generated 2500 shapes using our method on the Faust-10 dataset. The coverage of pose space can be studied by how well it performs on pose retrieval. Here, we sample shapes from SMPL, and for each sampled shape, we retrieve the closest shape generated by our method by L2 and compare it with the corresponding closest shape retrieved from the original set of landmarks. We illustrate these in the figure below, with triplets of query shape (blue), closest shape in Faust-10+GLASS (gold) and closest shape in original Faust-10 (grey).

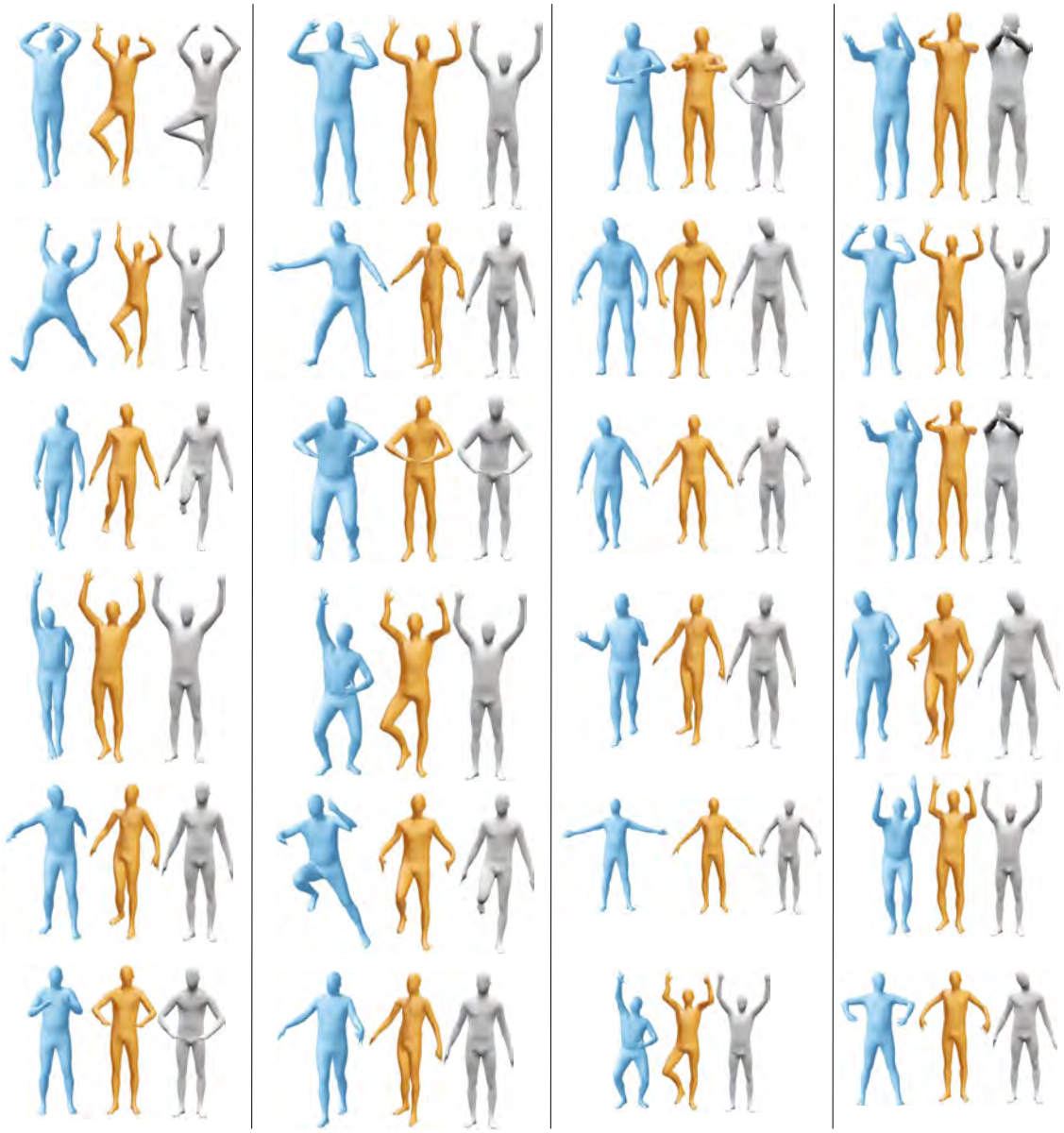

Figure 29: Each triplet shows a query shape from SMPL (blue) and the corresponding closest shapes in Faust-10+GLASS (gold) vs Faust-10 (grey).

9. Incremental training image

Here we show how adding new poses to the training landmarks changes the direction of pose-space exploration.

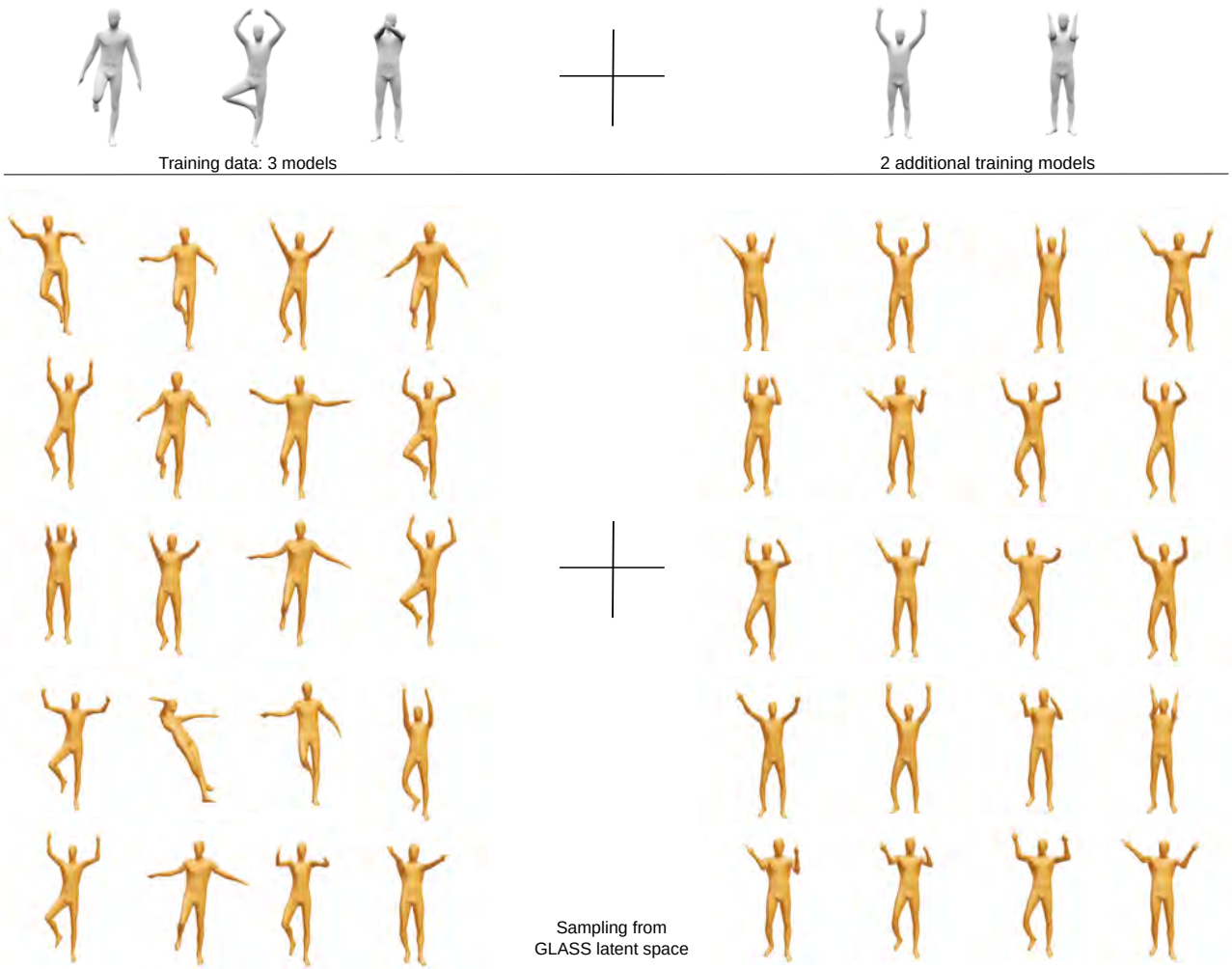

Figure 30: Effect of adding new poses to training set.

10. Network architecture

We use Pointnet encoder with convolutional layers and a multilayer perceptron decoder. Our VAE architecture is illustrated below.

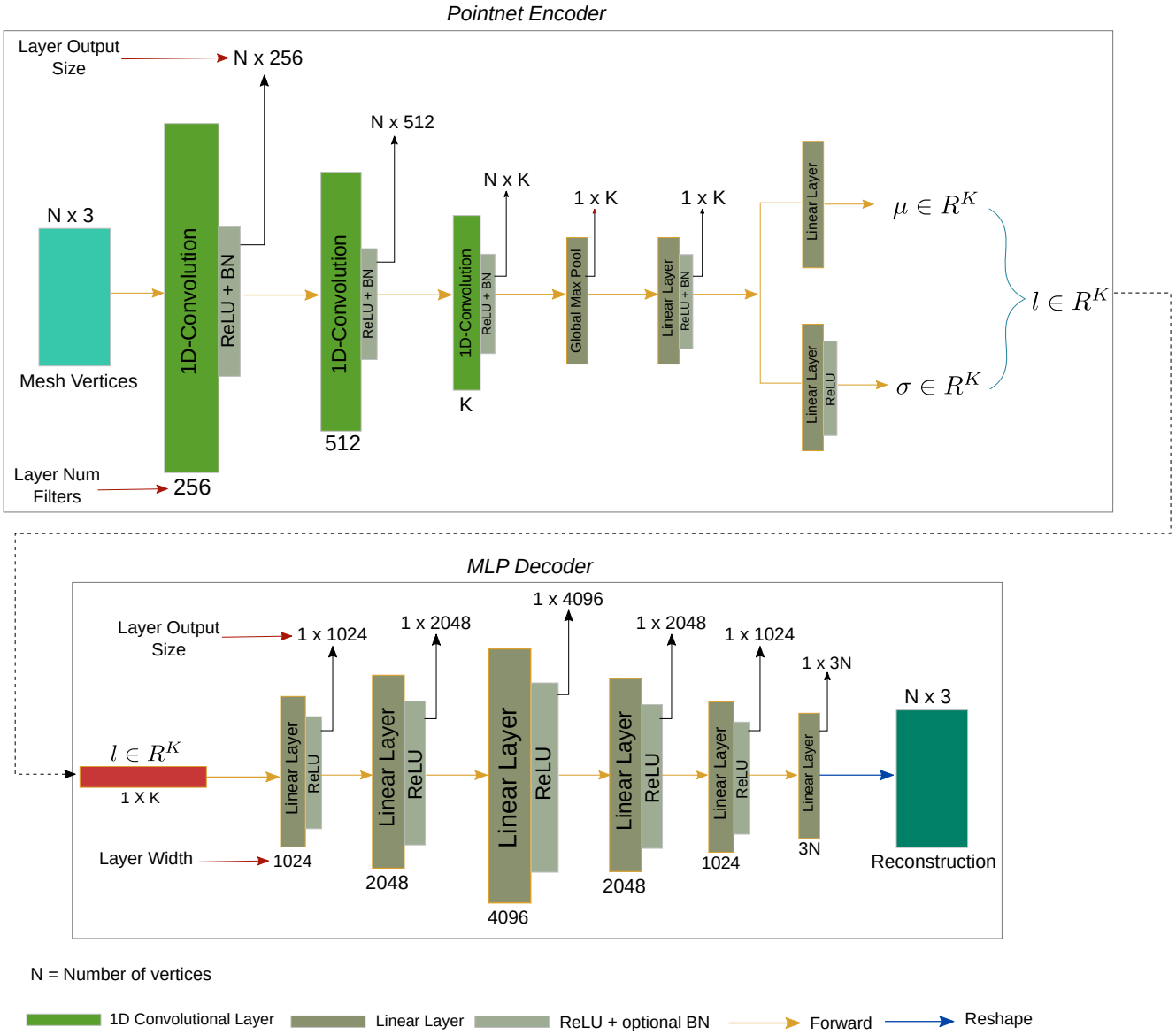

Figure 31: VAE network architecture: Pointnet Encoder (top) contains 3 convolutional layers (green) with ReLU/BN, followed by Global Max Pooling to obtain a feature for the set of vertices. The feature is input to a linear layer that further branches into 2 layers that output the gaussian parameters μ and σ from which we sample the latent code to be input to the decoder. The decoder is a 5 layer fully-connected network that transforms the latent code to a $3N$ size output which is reshaped to Nx3 to give us the reconstruction of vertices.

11. High Resolution results for Faust-10

While we used low resolution meshes of 1000 vertices for evaluation, our method scales well with number of vertices. Here we show samples generated when GLASS is trained on Faust-10 in its full resolution of 6890 vertices. The generated samples have low ARAP energies but can have non-semantic deformations as we do not use multiscale Laplacian features in this experiment.

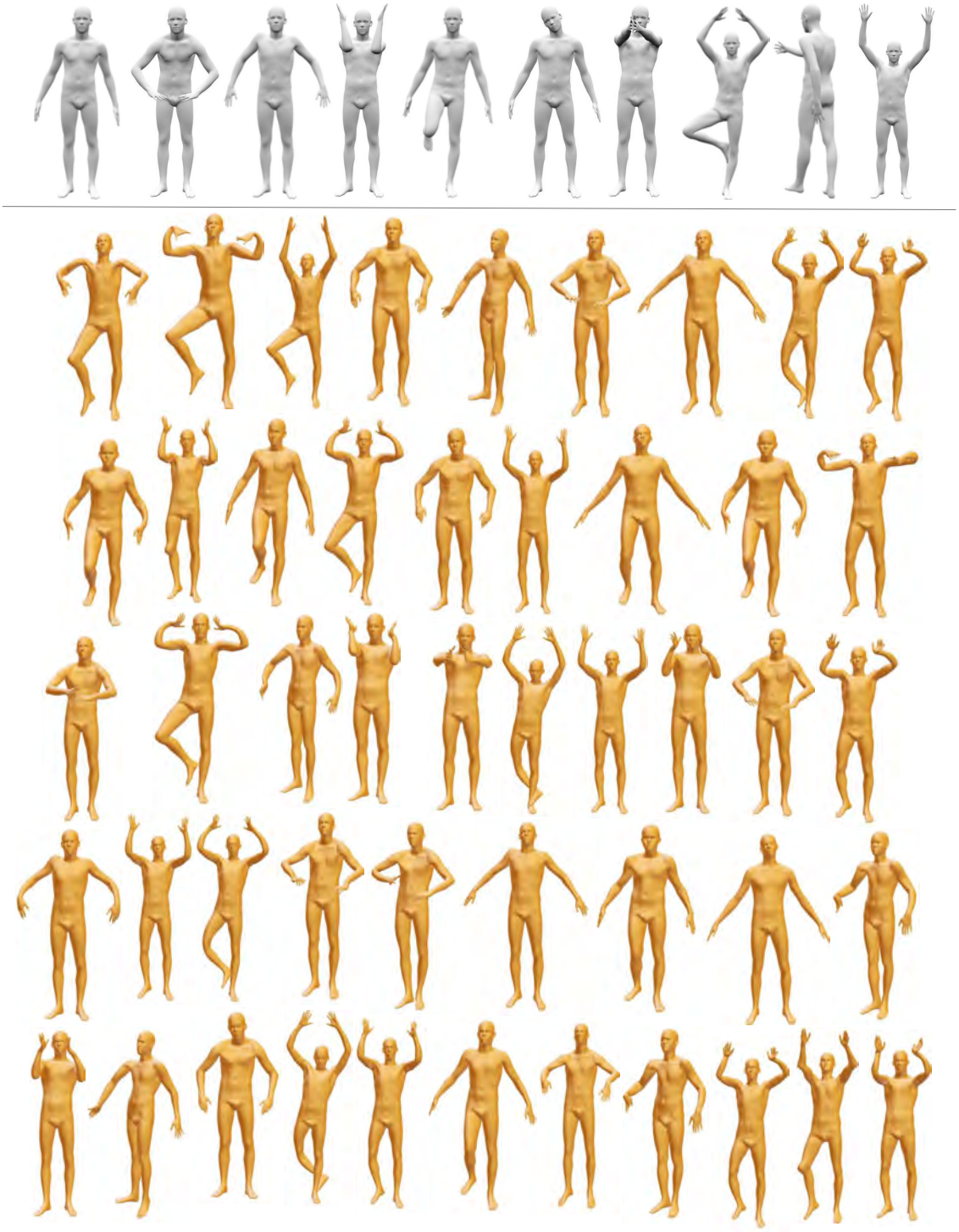

Figure 32: Samples generated with GLASS when trained on high resolution Faust-10.